Evaluating the latest Optane memory: A glorious swansong?

Adrian Jackson
a.jackson@epcc.ed.ac.uk
EPCC, The University of Edinburgh
Edinburgh, UK

ABSTRACT

In this paper we evaluate the 3rd generation of Intel's Optane non-volatile memory technology, assess the performance it can provide, and investigate the modes of use that can be beneficial in high performance computing, both for application performance and system architecture. We demonstrate sustained performance and functionality improvements from the latest hardware, along with I/O and memory performance and functionality that is not available from other memory or storage hardware. We show that leveraging Optane can provide significant reductions in the required volatile memory for applications, with minimal performance impacts, with appropriate memory hierarchy designs and considerations.

CCS CONCEPTS

• **Computer systems organization** → *Distributed architectures*; • **Hardware** → **Memory and dense storage**; **External storage**; • **Information systems** → **Distributed storage**.

KEYWORDS

Optane, NVRAM, B-APM, Storage, I/O

1 INTRODUCTION

Intel and Micron's 3DXPoint non-volatile memory, also known as Optane, was one of the most significant advances in memory technology of recent history. A product with much promise, providing both high I/O performance and persistent memory characteristics, it stood to bridge the gap between (relatively) small volatile memories, and large persistent storage hardware. With performance slower than DRAM, but higher than NVMe or SSDs, even when using high performance interconnects for storage devices, it had the potential to significantly change computing systems architectures and applications.

One of the key characteristics of Optane memory is its potential to live within the memory controller's domain, sitting on standard memory buses, albeit requiring modified memory controllers capable of dealing with heterogenous memory latencies within the same communication domain. This opened up byte-addressable functionality, extending I/O performance from block level operations (i.e. good performance on 4K contiguous read and write operations) down to cache line levels (i.e. 256 bytes).

The byte-addressable persistent memory (B-APM) functionality, coupled with high capacity, ushered in a period of true hierarchical memory, with systems available where RAM and B-APM were provided both under the control of the same processor(s), and the potential for future systems with high-bandwidth on-chip memory (HBM), as well as DRAM and B-APM, in the same memory domain. Unfortunately, it is not to be, or not to be yet, as first Micron then Intel have decided to discontinue manufacturing 3DXPoint memory, consigning it to the list of developed but unavailable hardware technologies.

Whether this was caused by Optane not live up to the original hype about potential performance, or because manufacturing costs and yields were not favourable for economical production and sales of Optane, is hard to tell without in-depth insider knowledge from the companies concerned. What is clear, though, is it has left a nascent ecosystem structured around hierarchical memory and exploiting multi-level memory for application data storage and I/O.

Future non-volatile memory is likely to be supported through disaggregated approaches, such as that developed by the Compute Express Link consortium (CXL)[1], where devices such as volatile and non-volatile memory, storage hardware, and accelerators, can be accessed transparently across network connections. However, whilst this approach has advantages around sharing resources across nodes and clusters, it does not provide the byte level performance that true in-node hierarchical memory, such as Optane plus DRAM, provides.

Whilst 3DXPoint memory has been cancelled, Intel recently produced the third generation of hardware products in this line, also known as Crow Pass, deployed using a DDR5 interconnect, and promising higher I/O performance than the previous two generations of Optane hardware. In this paper, we present results from benchmarking this third generation Optane memory, compare to the performance of first generation Optane memory, and discuss some of the potential benefits from using this type of memory within high performance computing (HPC) systems and for HPC applications.

Whilst this current generation of Optane memory will be the last for the foreseeable future, these explorations into hierarchical memory should still provide useful insight and illumination on how hierarchical memory can be exploited and utilised by applications and system designers, and help the computing community understand how disaggregated memory such as that provided by CXL devices, and hierarchical memory, such as that provided by Intel's Sapphire Rapids processor with onboard HBM and in-node DDR5 memories, can be best exploited and utilised.

2 OPTANE MEMORY

The B-APM memory that is provided in Optane DIMMs is a form of memory that is both non-volatile (i.e. data persists through power cycles) and can be accessed directly by the CPU through load and store operations. Intel's *Optane Data Centre Persistent Memory Module* (DCPMM for short) offering has been through three generations of technology, originally starting in a DDR4 socket compatible DIMM form factor, which has been recently upgraded to

[1] https://www.computeexpresslink.org/

Table 1: STREAM bandwidths with varying locations used for reading and writing data

Memory Location	Copy (GB/s)	Scale (GB/s)	Add (GB/s)	Triadd (GB/s)
All in DRAM	136	135	152	154
Read data in Optane Write data in DRAM	80	80	79	80
Read data in DRAM Write data in Optane	16.7	16.7	25.0	24.9
All in Optane	15.5	15.1	21.2	21.4

DDR5. These Optane DIMMs can be used alongside volatile memory DIMMs, using the same memory slots, meaning systems can support varied configurations of volatile and non-volatile memory spaces within the same compute node. Optane DIMMs are denser than volatile memory, meaning significantly more capacity can be provided through this form of hardware (typically 128, 256, or 512GB per DIMM).

To support the different memory types within the same memory domain, specialised memory controllers are required, necessitating specific Intel processors that support Optane memory. The extended memory controllers required additional functionality to deal with the varied latencies of the different forms of memory in the system, with Optane memory having both higher read and write latencies than volatile DDR memory, and with Optane asymmetric I/O performance (read operations having lower latencies than write operations, as demonstrated in the results presented in Table 1). This data is from a system using first generation Optane memory, with two 24-core processors in a single node, with 1.5TB of Optane memory per processor.

Optane DIMMs contain their own controller, which handles the aggregation of write or read operations, encryption, and other hardware level functionality. They can be operated in two modes, one where the non-volatile memory provided by the Optane DIMMs in a node is used as the main memory space, and any volatile DRAM is used as a last level cache for that main memory space (memory mode, or two-level memory), and one where volatile and non-volatile memory are both present as separate address spaces within the system, and can both be accessed independently by applications (app-direct mode, or one-level memory).

Memory mode requires no application modification or program changes to access the non-volatile memory, but does not enable the use of the persistence functionality that Optane provides (data is stored persistently, but there is no functionality to safely flush the DRAM used for last level cache to that persistent memory, so the consistency of data cannot be guaranteed). App-Direct mode allows applications to utilise Optane as persistent storage, but required programmers to either utilise the PMDK library (described in more detail in the next section) or to use the Optane memory as a block device (i.e. a standard file storage device).

In this paper we focus on the App Direct mode, as Memory mode operations do not fully exploit the functionality Optane provides, and performance is directly managed by the memory subsystem as with any cache functionality.

3 PROGRAMMING OPTANE MEMORY

The primary mechanism provided for programming Optane memory is the Persistent Memory Development Kit (PMDK) library [2]. This approach re-uses the naming scheme from filesystems as traditional persistent entities and maps the B-APM regions into the address space of a process (similar to memory mapped files in Linux). Once the mapping has been initiated, the file descriptor is no longer needed and can be closed. In this way, data can be retrieved from the B-APM using the file handle as the entry point to the memory address space, but I/O operations are undertaken using memory functionality, rather than file I/O operations. This bypasses the block I/O approach of traditional filesystems, removing both the asynchronous nature of operating system I/O calls and the associated I/O access sizes (block-based I/O, i.e. 4KB operations).

PMDK provides a range of different functionality, from memory pools to object stores, but for direct memory operation, the `libpmem2` library gives read/write functionality from application code. This `C` library provides functions to allocate regions of memory with the Optane B-APM, along with optimised calls to undertake standard operations (i.e. `memcpy`), and to **persist** data to B-APM.

As application data lives within the processor memory hierarchy, including multiple levels of cache on processors, data that has been created within B-APM using PMDK may not actually have been stored on the B-APM hardware at a particular point in time. When created and operated on, it stays within processor caches until evicted or otherwise flushed back to the B-APM. Some Intel systems provide some automatic flushing guarantees for B-APM, using ADR (Asynchronous DRAM Refresh) and eADR (extended Asynchronous DRAM Refresh) [1].

ADR utilises power supply functionality that can signal potential power loss events to the system with sufficient time that the latent power left within the system can be used to write back any data that is resting in the write pending queues on the processor's memory controller. This enables writes that are scheduled to be guaranteed once the memory controller has them, but puts significant responsibilities on programmers to ensure that updates are flushed in the correct order and time period to ensure they will be guaranteed to make it to the B-APM even if power is lost.

eADR extends this functionality to include the processor caches as well as the memory controller write queues, relaxing the consistency constraints imposed on programmers. With eADR any data still resident on the processor in a modified form will be flushed back to the B-APM when a power failure event is detected.

ADR and eADR have been developed to overcome a performance issue associated with programming Optane B-APM, namely the cost of ensuring data is persisted to the memory hardware. As standard programming memory operations do not consider the persistent nature of B-APM, i.e. they were designed with volatile memory in mind, there is no automatic mechanism for ensuring data has left the processor cache hierarchy and been updated on the non-volatile memory. PMDK provides a set of function call to do this, which enable a range of memory to be persisted to the B-APM, and ensure that after the function call has completed the data is persistently stored. These functions use processor instructions such as `CLWB`,

Table 2: STREAM bandwidths using first generation Optane memory with varying persist function call placements

Persist Location	Copy (GB/s)	Scale (GB/s)	Add (GB/s)	Triadd (GB/s)
No persist calls	15.5	15.1	21.2	21.4
Persist call at the end	13.9	13.9	19.7	19.7
Persist call after each operation	0.88	0.93	1.3	1.4

`CLFLUSH`, or `MFENCE`, to flush data out of caches and automatically ensure they end up at the Optane memory.

However, as they remove data from cache and wait until the data has reached the B-APM controllers on the DIMMs, these can be slow operations, and can disrupt the performance benefits that caches provide. Without ADR and eADR, programmers are responsible for manually adding persist function calls in their code at places that ensure data consistency should power be lost, but in a manner that reduces performance as little as possible.

Table 2 outlines the performance impact of different choices in the persist operation placement on the STREAM benchmark [7] utilising first generation Optane memory. We would note that the STREAM benchmark does not necessarily show the full performance impact of such operations as STREAM does not significantly reuse data within caches. The benchmark was run on a dual processor node with 24 core and 1.5TB of first generation Optane memory per processor.

Clearly, eADR functionality simplifies these programming constraints and is a big step towards improving performance for programs exploiting Optane memory. However, it requires this hardware functionality to be present in the systems hosting the Optane memory, both at the power support/systemboard level, and within the processor memory controller.

Had Optane not been discontinued by Intel, it is likely this would have become the de facto configuration for systems hosting Optane memory, however at present it cannot be guaranteed, so application development to safely exploit Optane memory still requires functionality to manually persist memory at points that ensure data consistency should a power loss occur whilst having a minimal impact on memory performance.

4 RELATED WORK

There has been a range of work looking at the performance of Optane memory since it debuted. [8] benchmarked a range of applications on first generation Optane memory, demonstrating improvements across all benchmarks when compared to standard I/O devices, but only modest performance improvements when integrated in scientific applications.

In [10], the authors propose system architectures to exploit first generation Optane memory and integrate this into HPC system architectures. They primarily focussed on using Optane for I/O, i.e. as a storage target, but demonstrated good performance improvements for large computational workflows and for applications utilising Optane in compute nodes for temporary storage. They also highlighted the work required to integrate such local storage into a shared computing environment.

The DAOS storage system [6] has shown the potential to leveraging Optane memory for small I/O operations, developing a persistent object storage system that has shown very high performance (including topping the IO500 list a number of times) by holding metadata and other small data objects within Optane memory. They have recently started the process of transitioning from Optane to DRAM for these operations [4], in absence of continued Optane product lines, but acknowledge that this will have performance impacts on DAOS for some usage scenarios.

Others have also investigated exploiting Optane for specific applications using prefetching, data placement, and customised persistent windows [9]. These have demonstrated that such hierarchical memory configurations can bring benefits over automatic or simplistic data placement approaches, but fall short of showing overall performance or system configuration benefits for applications through using hierarchical memory in general.

In this paper we investigate both I/O and memory performance for Optane, demonstrating the benefits in utilising this memory in both scenarios. We also explore the overall system benefits that using a hierarchical memory structure can provide. We provide details on how to program Optane memory from applications, as well as presenting, to the best of our knowledge, the first evaluations of third generation Optane memory performance for a range of benchmarks.

5 EVALUATION

We have used a range of benchmarks and approaches to evaluate the performance of Optane memory, both as an I/O device and as memory. Two different resources were used to benchmarking Optane memory, the first being a system that has dual socket Intel Xeon Platinum 8260M Cascade Lake processors, with 192GB of DDR4-2666 volatile memory, and 3TB of Optane series 100 memory. The second system has a single socket of Intel Xeon Platinum 8468 (Sapphire Rapids) with 48 cores, 128GB of DDR5-4800 volatile memory, and 2TB of Optane series 300 memory. Both systems used the Intel compilers (OneAPI versions), along with the Intel MPI library where required.

5.1 I/O Performance

To evaluate I/O performance we have utilised two distinct benchmarks. The first is the standard IOR benchmark [5], that measures bulk I/O performance in a variety of operation modes. We ran IOR on a single node of both systems we have access to, using the IOR easy node (i.e. single file per process), for a range of different I/O operation sizes (transfer size), writing a total of 2GB of data per process, or 96GB of data for a full node using 48 processes.

We ran two configurations of IOR, one using the POSIX file interface with the Optane memory mounted as an `fsdax` filesystem, the other using the PMDK library to write data directly as memory operations. Both of these use integrated IOR I/O providers, and have the same setup except for the I/O provider used. We tested I/O operations from 256-bytes, up to 16MB, with the results for both read and write bandwidth shown in Figures 1 and 2

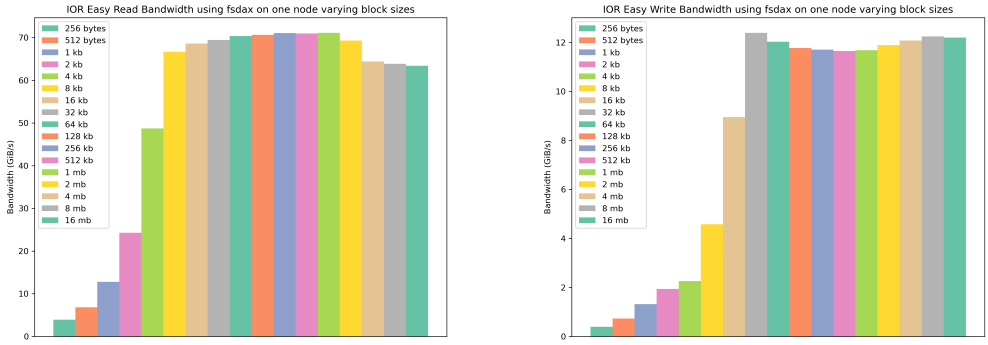

Figure 1: IOR Bandwidth using fsdax filesystem on a single node with third generation Optane

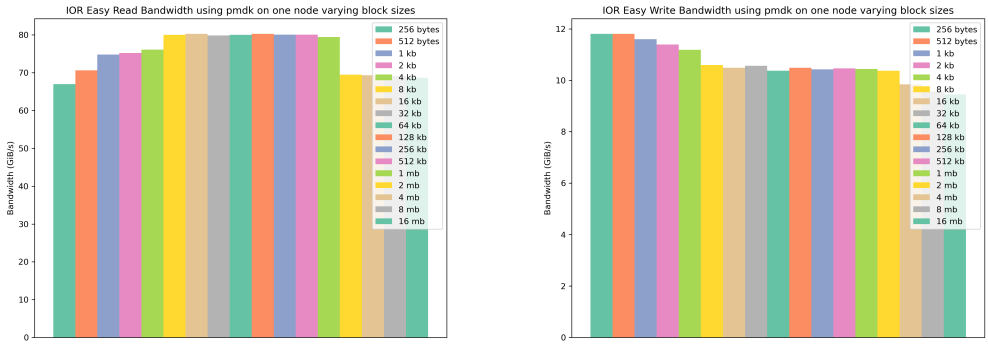

Figure 2: IOR Bandwidth using the PMDK library on a single node with third generation Optane

What is evident from the results presented is that both mechanisms for accessing the non-volatile storage that Optane provides, i.e. either as a filesystem or as memory, can provide high performance at large I/O operation sizes. We can see around 70GB/s of read bandwidth and 12GB/s of write bandwidth for both PMDK and fsdax, with PMDK providing slightly higher read performance for some transfer sizes (i.e. up to 80GB/s), and fsdax providing slightly higher write performance (just over 12GB/s).

We can also see the strong asymmetry of performance between read and write for Optane memory, with write bandwidth around six times lower than read bandwidth for the maximum achieved performances. However, the main performance result these figures illustrate is the different performance the interfaces give for small I/O operations. We can observe that using the B-APM as memory (i.e. through the PMDK interface) we see little to no reduction in performance for undertaking small data movements, both for reading and writing data. Indeed, the PMDK interface gives the best performance for writing at small data sizes, with a small loss of performance observed when scaling up to large I/O operations. This is likely to do with the sharing of the memory hardware between cores and saturation of the memory controller write queues on the Optane devices themselves when using larger I/O operation sizes.

We see very similar performance when undertaking the same benchmarks on first generation Optane memory, as highlighted in Figures 3 and 4. Similar I/O performance patterns are evident for this older Optane hardware, with traditional filesystem access requiring larger I/O operation sizes to get good sustained performance, and PMDK providing as good or better performance for small I/O operations compared to larger I/O operations.

We can also see that the newer Optane (the third generation hardware) provides improve performance for write, for filesystem operations (12GB/s vs 9GB/s) and better performance for large memory write operations (i.e. PMDK, ~11GB/s vs ~9GB/s). However, we do not see a significant improvement in peak writing performance for these configurations for the newer version of Optane memory. It could be that this is due to the mode of operations IOR is running in, i.e. 48 processes per node, with high levels of contention for the Optane memory.

On the read side, again, performance is improved for smaller I/O operations for filesystem interfaces, with good read performance sustained down to 8KB transfers on fsdax. Better read performance is also observed for larger transfer sizes (i.e. 2MB+) for the fsdax interface with this newer Optane memory. Similar trends can be seen for the PMDK interface on third generation Optane, with

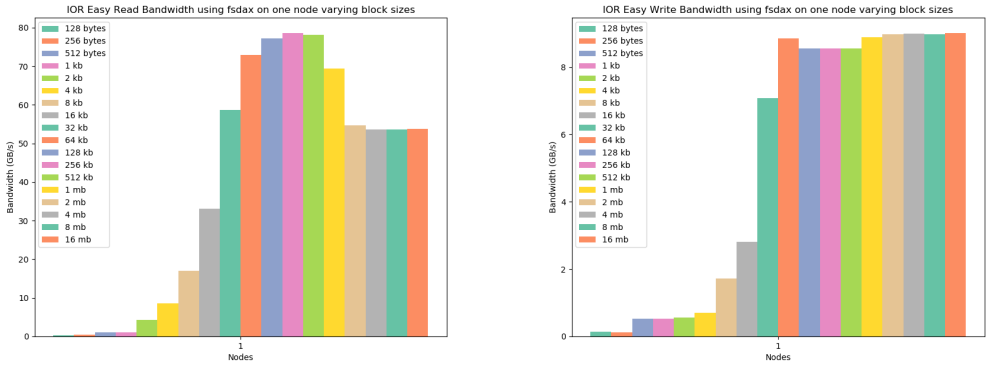

Figure 3: IOR Bandwidth using fsdax filesystem on a single node with first generation Optane

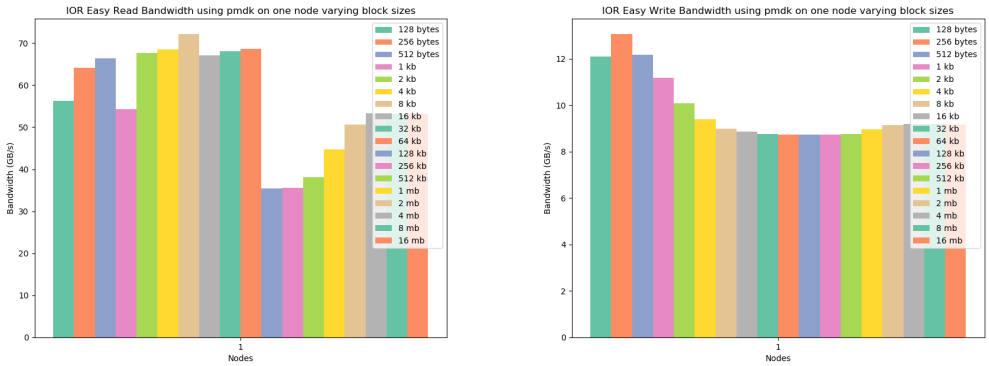

Figure 4: IOR Bandwidth using the PMDK library on a single node with first generation Optane

sustained read performance being maintained up to 2MB transfer sizes, whereas this dropped off significantly at 128KB transfer sizes for first generation Optane.

The second I/O benchmark we used was MADBench2 [3], a synthetic I/O benchmark designed to mimic application I/O from a large-scale cosmic microwave background (CMB) data analysis package. This has more small and varied I/O operations, and mimics out-of-core or out-of-memory style operations (where the dataset is too large to be stored in volatile memory, so it kept in storage and processed in chunks as required). This is an ideal benchmark to assess Optane memory usage for I/O because it enables active evaluation of a hierarchical memory configuration, emulating the situation where a larger dataset is stored in slower but larger memory and the active part is hosted in the faster, volatile, memory.

As implemented, MADBench2 uses files to store the out-of-core data, so we extended this benchmark to utilise PMDK and thereby store the out-of-core data in Optane memory instead. This functionality, which we could classify as *out-of-volatile* data, enabled us to benchmark Optane through PMDK vs Optane through fsdax, and explore how I/O vs memory operations affect performance for such situations. We also modified the MADBench2 benchmark to allow out-of-volatile data to be done at a range of block sizes, rather than the single fixed block size originally implemented.

As Figure 5 shows, the performance comparison between fsdax filesystem accesses and PMDK memory accesses for MADBench2. The two graphs show the same results, except the graph on the left has excluded the very slow results (Original(Blocked:8)) to make the performance of the rest of the results more visible. The error bars on the graph show the range of variation in performance for the benchmark (each benchmark was repeated 5 times, with the bars show the mean duration).

We can see that the benchmark on fsdax gives similar, if not quite as fast, performance as PMDK memory accesses in most cases. The blocking of the dataset to a finer granularity has no performance impact for the PMDK implementation, mirroring the results previously seen with IOR. Blocking for the filesystem brings some small performance benefits for large size blocks (the standard block size is 128 elements in each dimension), but brings very significant performance reduction for small block sizes (8 elements in each dimension).

This succinctly demonstrates the potential for B-APM like Optane to provide a hierarchical memory structure that can enable

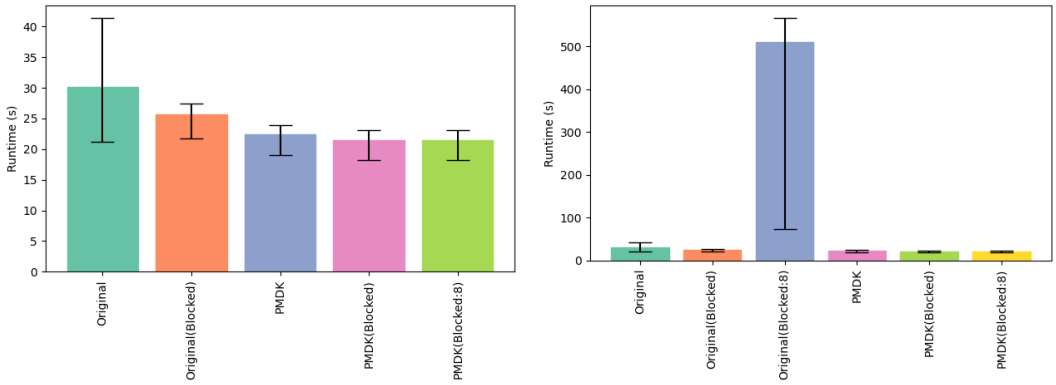

Figure 5: MADBench2 performance on third generation Optane

reduced volatile memory capacity whilst still maintaining performance compared to storing large amounts of active data in volatile memory. However, we can see in this configuration, we're required to move away from *out-of-volatile* functionality being an I/O implementation and towards it being memory-based to enable such functionality to be efficiently utilised.

5.2 Memory

We have already discussed some memory style benchmarking of Optane, both in the introductory sections, where we introduced the performance of first generation Optane, and in the MADBench2 benchmarking.

As part of evaluating the third generation of Optane memory, we have run the STREAM memory bandwidth benchmark on that hardware, and present the results in Table 3. These results show a good improvement in memory bandwidth compared to the equivalent first generation benchmark results, with an aggregate bandwidth of between 24 and 28GB/s depending on the specific benchmark been undertaken (all the data in B-APM). This is a performance improvement of between 25-35%, and is even slightly higher than the DRAM performance improvements we see between the two systems used (DDR4-2666 to DDR5-4800).

The results also show that the latest Optane does not suffer a performance overhead for minimal persist operations (i.e. there is no significant performance difference between not persisting data and persisting it at the end of the benchmark). This is an improvement over the first generation Optane results. Furthermore, the results for high levels of persistent operations (persist call after each operation) also show significant improvements, with a ~3x increase in bandwidth.

This demonstrates the improvements that have been achieved through the new generations of Optane memory, reducing some of the overheads with persisting data to the memory hardware, sustaining performance under contention, and generally improving the read and write performance of the memory.

We have also run two benchmarks that mimic application kernels on the Optane memory, evaluating the performance impact and/or benefits of porting applications to a multi-tiered memory configuration and using B-APM for part or all of the data used by an application.

Table 3: STREAM bandwidths with varying locations and persistent horizons used for reading and writing data on Third Generation Optane

Memory Location (persist horizon)	Copy (GB/s)	Scale (GB/s)	Add (GB/s)	Triadd (GB/s)
All in DRAM	172	172	187	186
Read data in Optane Write data in DRAM	135	135	112	112
Read data in DRAM Write data in Optane	29	29	41	41
All in Optane (no persist)	23	24	28	28
All in Optane (persist at end)	23	23	28	28
All in Optane (persist each write)	3	3	4	4

The first benchmark, `sharpen`, is an edge detection kernel that sharpens an image by convolving the source image with a filter function. The convolution filter consists of both a simple denoising step using a Gaussian operator, and a Laplacian operator that generates edges from the input pixels using a second order derivative. These are discretised using standard n-sized stencil across a two dimensional data structure, with a range of arrays storing the original image, the updated image, and the convolutional filter. The application is parallelised with MPI enabling scaling to multiple nodes, and thereby to large image sizes.

The second benchmark, `cfd`, is an idealised computational fluid dynamics kernel, implementing a fluid flow simulation in a 2d box, with a single inlet and a single outflow. The program calculates the velocity and vorticity of the flow using finite viscosity, with a basic Jacobi discretisation, again using an n-sized stencil. As with the sharpen application, this has been parallelised with MPI and can scale up to large numbers of processes and thereby simulate large domains.

Table 4: Performance of applications ported to Optane memory

Application	Hardware	Runtime (seconds)	Volatile Memory (GB)
sharpen	DRAM	56	285
	DRAM+B-APM	63	170
cfd	DRAM	7.95	40
	DRAM+B-APM	9.64	25

Table 5: Performance of the CFD application

Persistency Horizon	Hardware	Runtime (seconds)	Volatile Memory (GB)
N/A	DRAM	7.95	40
No persist	DRAM+B-APM	9.64	25
Partial persist	DRAM+B-APM	10.67	25
Full persist	DRAM+B-APM	41.84	2

Both of these applications have a number of large arrays storing the input data and the data used for calculations during the execution of the program. When porting to utilise B-APM, the challenge is in deciding which arrays to implement on the B-APM and which to leave in volatile memory. Both applications have large arrays that are utilised to hold the source data for the simulation, but are not updated after initialisation. We chose to port these arrays to B-APM using PMDK functionality, and to leave the data that is frequently updated in volatile memory.

For the sharpen benchmark, we can demonstrate performance of around 5% less for a reduction in around half the required volatile memory. Likewise, for the cfd application, we are able to reduce the amount of memory required by around half, whilst having only a small impact on the overall performance of the program, as outlined in Table 4

We also investigated the cfd benchmark in more detail, with results outlined in Table 5. Here we investigated different persistent horizons, from no persist calls at all, through to persisting every data write. As reflected in the STREAM benchmark results, high volumes of persist calls significantly impact performance, but less frequent persistence calls have minimal impact on the overall application performance whilst still maintaining sensible levels of application coherency.

We can also see that moving to a full persistence model it is possible to reduce the residence set size required for the application, from 40 or 25GB down to 2GB. This involves only holding part of the active data in volatile memory at any one time, and swapping those active parts as the simulation iteration progresses. Whilst this had a ~4 fold reduction in performance, it did enable a 20 fold reduction in volatile memory requirements.

6 SUMMARY AND FUTURE WORK

We have evaluated the latest, and last, generation of Optane memory, and demonstrated that it bring improved functionality and better performance in some scenarios, although the performance is in line with the performance improvements expected for new generations of any memory technology.

We demonstrated the unique role Optane had the potential to play in computing architectures, that of enabling very small I/O operations with the same performance as bulk streaming I/O. This could have opened up new computing architectures, with significantly smaller volatile memory footprints, and a hierarchy of memory that could be exploited by applications designed to be aware of that hierarchy and how to efficiently exploit it. Reducing volatile memory would in turn allow for the reduction of both energy and power requirements for computing systems, thereby enabling denser, more efficient hardware configurations.

B-APM coupled with on-processor high bandwidth memory could have enabled the complete removal of DRAM whilst still retaining significant memory capacity. One architecture that has already removed in-node DRAM is the A64FX processor in the Fugaku system, where each processor has high bandwidth memory on chip but no associated DRAM. This has helped build one of the largest supercomputers in the world, but does mean that applications that require large amounts of memory per process necessarily have to scale up to large numbers of nodes (and potentially underpopulate those nodes) to be able to run. A well structured and programmed B-APM capacity within compute nodes would negate these issues.

With the cancellation of Optane, the proposed replacement technology is composable infrastructures, such as those exploiting the CXL architecture, which allow the attachment of non-volatile resource across networks transparently. CXL enable memory and storage targets to be defined, potentially replacing the persist memory that Optane provided. However, whilst high bandwidth and low latency CXL hard will become available, because of the mechanism for connecting composable resources (i.e. across some form of network), it will not match the byte-addressable, and therefore the small operation, performance and functionality that Optane provides.

Nevertheless, there is clearly scope to explore out-of-volatile implementations on composable hardware to map out the performance that is possible and the types of programs or algorithms that will be able to efficiently exploit them, and the benchmarks we have developed and deployed during this research will provide a good starting point for that research.

ACKNOWLEDGMENTS

Benchmarking results on first generation Optane hardware in this paper were run on the NEXTGenIO system, funded by the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 671951. Third generation Optane benchmarking was undertaken on the Tsukuba University Pegasus system, https://www.ccs.tsukuba.ac.jp/eng/supercomputers/, provided under collaboration between CCS, University of Tsukuba and EPCC, University of Edinburgh. Some of the research in this paper was supported by UKRI, through EPSRC grant EP/T028351/1.

REFERENCES

[1] 2023. *Extended Asynchronous Dram Refresh.* Retrieved August 20, 2023 from https://www.intel.com/content/www/us/en/developer/articles/technical/eadr-new-opportunities-for-persistent-memory-applications.html

[2] 2023. *Persistent Memory Development Kit.* Retrieved August 20, 2023 from http://pmem.io/pmdk/

[3] Julian Borrill, Leonid Oliker, John Shalf, Hongzhang Shan, and Andrew Uselton. 2009. HPC global file system performance analysis using a scientific-application derived benchmark. *Parallel Comput.* 35, 6 (2009), 358–373. https://doi.org/10.1016/j.parco.2009.02.002

[4] Michael Hennecke, Jeff Olivier, Tom Nabarro, Liang Zhen, Yawei Niu, Shilong Wang, and Xuezhao Liu. 2023. DAOS Beyond Persistent Memory: Architecture and Initial Performance Results. In *High Performance Computing*, Amanda Bienz, Michèle Weiland, Marc Baboulin, and Carola Kruse (Eds.). Springer Nature Switzerland, Cham, 353–365.

[5] Julian Kunkel, Glenn K. Lockwood, Mohamad Chaarawi, Christopher J. Morrone, Jean-Yves VET, otatebe, shanedsnyder, Rob Latham, Adrian Jackson, Wei keng Liao, Jeff Inman, Brett Kettering, Enno Zickler, jschwartz cray, Axel Huebl, Mark Nelson, Nathan Hjelm, Pablo Llopis, VinsonLeung, Christian Wassermann, Blair Crossman, adilger, Pidad D'Souza, Sylvain Didelot, jhendersonHDF, Adam Moody, Alfred Torrez, 川島和津実, John Bent, and Oliver Steffen. 2023. *hpc/ior: IOR version 4.0.0 release candidate 1.* https://doi.org/10.5281/zenodo.7662913

[6] Zhen Liang, Johann Lombardi, Mohamad Chaarawi, and Michael Hennecke. 2020. *DAOS: A Scale-Out High Performance Storage Stack for Storage Class Memory.* 40–54. https://doi.org/10.1007/978-3-030-48842-0_3

[7] John D. McCalpin. 1995. Memory Bandwidth and Machine Balance in Current High Performance Computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter* (Dec. 1995), 19–25.

[8] Vladimir Mironov, Igor Chernykh, Igor Kulikov, Alexander Moskovsky, Evgeny Epifanovsky, and Andrey Kudryavtsev. 2019. Performance Evaluation of the Intel Optane DC Memory With Scientific Benchmarks. In *2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC).* 1–6. https://doi.org/10.1109/MCHPC49590.2019.00008

[9] Jie Ren, Jiaolin Luo, Ivy Peng, Kai Wu, and Dong Li. 2021. Optimizing Large-Scale Plasma Simulations on Persistent Memory-Based Heterogeneous Memory with Effective Data Placement across Memory Hierarchy. In *Proceedings of the ACM International Conference on Supercomputing* (Virtual Event, USA) *(ICS '21).* Association for Computing Machinery, New York, NY, USA, 203–214. https://doi.org/10.1145/3447818.3460356

[10] Michèle Weiland, Holger Brunst, Tiago Quintino, Nick Johnson, Olivier Iffrig, Simon Smart, Christian Herold, Antonino Bonanni, Adrian Jackson, and Mark Parsons. 2019. 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* (Denver, Colorado) *(SC '19).* Association for Computing Machinery, New York, NY, USA, Article 76, 19 pages. https://doi.org/10.1145/3295500.3356159