NVM-based energy and cost efficient HPC clusters

Onkar Patil[1], Latchesar Ionkov[2], Jason Lee[2], Frank Mueller[1], Michael Lang[2]
[1]North Carolina State University, USA
[2]Los Alamos National Laboratory, USA
opatil@ncsu.edu,mueller@cs.ncsu.edu,{lionkov,jasonlee,mlang}@lanl.gov

ABSTRACT

Non-Volatile Memory (NVM) is a byte-addressable, high capacity, high latency and a persistent form of memory that can extend the primary memory hierarchy by another level. It is up to 8x denser than DRAM, allowing for clusters with compute nodes that have significantly higher memory capacity than those of previous generations. Intel's Optane DC Persistent Memory Module (PMM) is such an NVM device that can be used to increase the memory density of high performance computing (HPC) systems. This work hypothesizes that with higher memory density, scientific computing applications with larger problem sizes can be run on fewer compute nodes than on current HPC systems. This, in turn, can reduce operational cost. This work tests this hypothesis by comparing performance and energy of HPC jobs with large problem sizes of (1) fewer nodes with large NVM capacity under various configurations and (2) more nodes with an equivalent amount of DRAM memory. In experiments, performance and energy consumption are shown to be dependent on application characteristics: Codes optimized for high cache reuse suffer no performance degradation on NVM, combined with significant energy savings at lower acquisition and operational costs compared to traditional HPC systems without NVM. In contrast, memory bound applications using DRAM as a cache for NVM may provide a small performance benefit over using a DRAM-NVM hybrid memory.

CCS CONCEPTS

• **Computer Systems Organization** → **Architecture**; • **Computing methodologies** → **Massively parallel and high performance simulations**;

KEYWORDS

Optane DC, NVM, persistent memory, heterogeneous memory, Energy, Power, HPC

ACM Reference Format:
Onkar Patil[1], Latchesar Ionkov[2], Jason Lee[2], Frank Mueller[1], Michael Lang[2]. 2021. NVM-based energy and cost efficient HPC clusters. In *The International Symposium on Memory Systems, Oct., 2021, Virtual Conference.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3488423.3488440

MEMSYS21, Oct., 2021, Virtual Conference

ACM ISBN 978-1-4503-8570-1…$15.00
https://doi.org/10.1145/3488423.3488440

1 INTRODUCTION

HPC systems have been instrumental in running large-scale simulations of scientific problems in almost every field. These scientific applications require large compute and memory resources, e.g., the Summit supercomputer [11] at Oak Ridge National Laboratory (ORNL), to perform the simulations within reasonable amounts of time and with sufficient accuracy. According to the latest TOP500 list [30], we are currently within the petascale era of compute capability (PetaFLOPS), where problem sizes run into petabytes (PBs). By 2021, we expect to achieve exascale compute capability which will increase the problem sizes due to more complex simulations including whole slide image analysis [3, 27]. To run these massive simulations and analysis, along with exaFLOP compute capability, we also need a large amount of memory resources.

HPC systems today are large clusters of nodes [12] with compute and memory resources. The compute resources include the primary CPU and often additional co-processors or accelerators such as GPUs [16]. Memory is usually comprised of DRAM but recently accelerators have included separate high bandwidth DRAM (HBM) modules. The compute and memory resources across nodes are connected to each other with high-speed switches and interconnects. The clusters also have cooling and power equipment.

According to the TOP500 list released in June 2020 [30], the 10 fastest supercomputers in the world have a large number of nodes, which deliver 27 to 513 peak petaFLOPS and have main memory ranging between 200 TB to 4.85 PB. Their peak power consumption ranges from 1.5 to 28 MW. The total number of compute nodes in these clusters are in the multiples of thousands. Due to the use of accelerators in recent years, the number of compute nodes in a cluster has reduced with the increase in compute density per node. DRAM based main memory has scaled, too, but only at *half* the rate of increase in compute capability [19].

In order to achieve exascale capability, memory density needs to increase at the same rate as core count to solve larger problems. The Frontier supercomputer that will be operational in 2022 has an expected aggregate system memory of 10 PB [8] with approximately 6,400 nodes and a power envelope of 30 MW. Its acquisition cost is $600 million. However, if only traditional DRAM memory is used then memory density of the Frontier machine will not see any significant increase compared to Summit. This will restrict the problem sizes that can be run on the Frontier machine and reduce the ability to achieve results with higher accuracy at higher resolution and analysis with finer granularity. We focus on increasing the memory density of compute nodes and memory per core ratio in future HPC systems to support such large problem sizes with cost and energy efficiency.

NVM is a byte-addressable, high-density memory that can help increase the amount of memory in a single node of a cluster [20]. NVM can be produced using various technologies like Phase Change

Table 1: Comparisons of related work (BW: memory bandwidth, L: latency, ET: exec. time, EN: energy, CB: cache behavior, MN: multi-node)

Publication	Characteristics	Hybrid Memory	Allocation Policy	Applications	Problem sizes	Cost Analysis
Our work	BW ET EN MN	Yes	**Memory agnostic**	AMG,LULESH,VPIC	Large	**Yes**
M. Ben et al. [22]	BW ET	Yes	Memory aware	AMG,LULESH,SNAP,QMCPACK	Large	No
Patil et al. [23]	BW ET EN CB	Yes	Memory aware	AMG,LULESH,SNAP,VPIC,Custom	Small, Medium	No
Izraelevitz et al. [13]	BW L	Yes	Memory aware	SPEC 2006,2017,PARSEC	Small	No
Peng et al. [25]	BW EN L	Yes	Memory aware	GAP	Large	No

Memory (PCM) [15], spin-torque transfer RAM (STT-RAM)[1] or Resistive RAM (Re-RAM). NVM has a lower power consumption, as constant refreshes to maintain state are not required. However, all of these memories are slower than DRAM in terms of access latencies. Intel has been first-to-market with a NVDIMM form factor that can be used in conjunction with DRAM. The NVM device is based on PCM technology known as Optane DC PMM. The Aurora supercomputer will also have support for Optane DC PMM. It is eight times denser than DDR4 DRAM but is also approximately six times slower in terms of write latency [13, 23]. Also, Optane DC is cheaper than DDR4 in terms of cost per GB of memory by a factor of 1.5 to 2 depending on the capacity of the DIMM [21].

Contributions: This work evaluates the hypothesis that using NVM devices like Intel's Optane DC PMM as an extension to the existing memory hierarchy can support larger problem sizes in fewer number of nodes, keeping cost and power consumption within a target budget, compared to DRAM-only nodes. The slower write speeds of NVM and lower aggregate compute capabilities over all nodes may reduce the performance of the system, but the reduced inter-node communication over the interconnect, increased memory per core ratio, reduced acquisition cost and energy efficiency provide be a trade-off for the slowdown [10]. The increased memory size that NVM can bring to a node will result in maintaining the required overall memory capacity across a cluster with fewer number of nodes. This hypothesis is tested by measuring the performance characteristics and power consumption of compute nodes with and without NVM by executing large problem sizes of a set of HPC applications (VPIC, AMG and LULESH).

Recent works [13, 23] characterize the performance of Optane DC for different HPC workloads and benchmarks compared to DRAM. However, they lack the performance characterization of a multi-node system with a DRAM-NVM based hybrid memory address space where the application allocates data structures on both DRAM and NVM for a given execution combined with a comparison to the same execution on a traditional DRAM-based HPC systems. Our work fills this gap and provides novel insights for running applications on large memory nodes. It also provides the baseline comparison for procuring future HPC systems while considering different memory systems.

2 RELATED WORK

Table 1 provides a feature comparison of our work to recent studies that characterize the performance of NVM technologies especially with Intel's Optane DC PMM.

Izraelevitz et al. [13] evaluated the memory access characteristics of Optane DC PMM for different file-systems, database applications and performance benchmarks. They found that Optane DC boosts the performance of file-systems and database applications due to lower latencies than storage devices. In contrast, we focus on using Optane DC as an extended memory address space for the existing memory hierarchy for HPC applications. Patil et al. [23] characterized the performance of a DRAM-NVM hybrid memory system for HPC applications. They measured the performance of frequently occurring HPC kernels and HPC applications executing on Optane DC with executions on DRAM and executions with DRAM as cache for Optane DC. They compared the performance of all memory devices whereas we are measuring the performance of two different memory systems. Peng et al. [24, 25] evaluated the Optane DC PMMs in all the configurations available and also measured the performance of separating reads and writes on a DRAM-NVM memory system. Zivanovic et al. [36] evaluated scaling-in; i.e. decreasing the number of application processes and compute nodes to solve a fixed-sized problem, using a set of HPC applications running in a production system. Our work focuses on comparing the performance of memory system agnostic executions of HPC applications on DRAM-NVM and DRAM-only memory systems with a flat address space. Our evaluation spans over multiple nodes with DRAM-NVM hybrid memory space whereas other works have evaluated only the single node performance.

Recent studies focus on creating systems and policies to enable hybrid memory architectures. M. Ben et al. [22] used profiling information obtained from PEBS based characterizations to guide memory allocations on complex memory systems. Zhou et al. [34] proposed a victim-aware cache policy to improve the lifetime of NVM in a hybrid memory system. Rodriguez et al. [28] examined write-aware replacement policies of data in PCM-based systems. Ma et al. [17] implement an asymmetric NVM architecture where the byte-addressable NVM is decoupled from the compute nodes and a framework enables full data structure replication using operational logging of all read and writes. This can limit the problem sizes. In contrast, our work focuses on improving the memory-per-core ratio in the compute nodes by utilizing the NVM as a byte-addressable memory expanding device which increases the support for large problem sizes. There have been many studies to identify challenges related to achieving exascale systems. Bergman et al. [4] detailed numerous challenges that need to be overcome to reach exascale HPC capability. They outlined the memory challenges the current systems face and survey many options to overcome them. They also examine the challenges related to capping the power of exascale systems. Peterka et al. [26] identified similar challenges to achieve exascale computing and focused on memory bandwidth and capacity issues and how to mitigate them. Ashraf et al. [2] examined methods of improving HPC performance under desirable power caps. Gamatié et al. [9] empirically surveyed NVM technologies

for energy efficient HPC systems. In contrast, we focus on evaluating available technology to increase the memory density and keep power consumption and operating costs of HPC systems down.

3 BACKGROUND

Supercomputer architecture has been constantly evolving over the last six decades. These changes were driven by the need to solve larger and more complex problems, technological advances and cost justification. In the 1970s, supercomputers were leading edge processor machines specially designed to perform floating point operations as fast as possible [16]. However, within a decade, supercomputers were built using multiple processors connected by different means as the cost of building faster processors rose astronomically. This eventually led to the development of cluster computing, where commodity hardware was connected over a fast network (e.g., Beowulf Clusters). The clusters were further sped up by introduction of multi-core processors, which were designed to reduce the speed gap between memory and processors, increase instruction level parallelism and improve power efficiency. A decade ago, GPUs were introduced in clusters in order to utilize Single Instruction Multiple Data (SIMD) parallel processing and boost the performance of the supercomputers. Today, large clusters with processors and GPUs, connected by a high speed interconnect, are leading the TOP500 list of supercomputers.

Memory architecture also went through changes along with the changes in supercomputer architecture [20]. With monolithic supercomputers, the memory system was simple with a single type of main memory. As the processing speeds increased, more memory was required to support the large problems that were executed on the faster processors. However, memory access speeds did not increase at the same rate as processor speeds. This led to the introduction of different memory technologies that are used as buffers or caches to reduce the performance impact of larger but slower memory. The memory architecture was split into different tiers, from faster but smaller memory to slower memory with massive capacities, in order to support multi-core processors and parallelism. This has created the memory hierarchy we have today.

The changes in supercomputer architecture were also sparked by the evolution of different or more efficient numerical solvers and methods for scientific problems over the decades as well as development of software that enabled the efficient use of new hardware. Supercomputers eventually became very specialized, i.e., to solve only certain types of problems efficiently [12]. The Summit supercomputer is very efficient in running artificial intelligence and big data problems, which are predominantly SIMD-based problems. It may not be the best suited platform for other types of problems.

Memory technology has not scaled well in terms of capacity compared to the scaling of processing speed [19]. This has led to inefficiency in the design of clusters as the memory capacity is critical in order to run large problem sizes. The system designer has to strike a balance between the number of nodes in a cluster, the amount of memory, and the size and speed of the interconnect when given a specific cluster cost, power budget and target performance. Due to inefficient scaling of DRAM capacity, a large number of nodes will be required in a cluster to support the target problem sizes at exascale. This will also add more communication infrastructure to the cluster, which acts as a performance bottleneck and consumes significant amounts of power. We hypothesize that the increased memory capacity that NVM can bring to a node will result in maintaining the required memory capacity of the cluster with fewer number of nodes. This will help in solving large problem sizes while staying within target cost and power budgets.

4 ARCHITECTURE

To test our hypothesis, we set up an experimental architecture with actual servers. We have a single node with DRAM-NVM hybrid memory and 4 nodes with only DRAM memory that can support the same problem size. The 4 nodes are connected to each other using an Infiniband switch. We also used a setup of 2 DRAM-NVM hybrid memory nodes connected over Infiniband to measure and isolate the effect of communication on the performance of applications. The specifications of each node are described in Table 2.

Table 2: Experiment Platforms

Specifications	**Optane Node**	**DRAM Node**(x4)
Model name	Intel Xeon 8260L	Intel Xeon 6152
Architecture	x86_64	x86_64
CPUs	96	88
Sockets	2	2
Cores per socket	24	22
NUMA nodes	4	2
CPU MHz	3100	2900
CPU max MHz	3900	3700
CPU min MHz	1000	1000
L1d cache	32 KB	32KB
L1i cache	32 KB	32KB
L2 cache	1 MB	1 MB
L3 cache	35.3 MB	30.25 MB
Memory Controllers	4	2
Channels/controller	6	6
DIMM protocol	DDR4	DDR4
DRAM size	192 GB	384 GB
Max. DRAM BW	104 GB/s	104 GB/s
NVDIMM protocol	DDR-T	None
NVRAM size	1.5 TB	None
Max. NVRAM BW	40 GB/s	None
No. of nodes	2	4
Interconnect (BW)	Yes (100 GB/s)	Yes (100 GB/s)
Avg. Power Consumption	440 W	330 W
Operating System	CentOS 7	CentOS 7
Acquisition Cost factor	1	0.65

We refer to the nodes with NVM as the Optane nodes. In the Optane nodes, we have two sockets, each with Intel's 24 core Cascade-Lake processor with hyper-threading turned on, which effectively provides 96 processing units. Each core has a 32 KB private L1 instruction cache, a 32 KB private L1 data cache, and a private 1 MB L2 cache. There is a 35.3 MB L3 cache shared between all cores. Each socket has 12 DIMM slots. 6 of the slots are occupied by 16 GB DDR4 DRAM modules and the other 6 slots are occupied by 128 GB Optane DC modules. This adds up to 192 GB of DRAM and 1.5 TB of non-volatile memory. The nodes have 4 memory controllers in total. Two of the memory controllers are connected to 6 DRAM DIMMs each, and the other two, known as iMC, are connected to 6 NVDIMMs each.

The nodes without NVM are referred to as DRAM nodes. Every DRAM node has two sockets of Intel's 22 core Skylake processor with hyper-threading turned on, which effectively gives 88 processing units. Each core has a 32 KB private L1 instruction cache, a 32 KB private data cache, and a private 1 MB L2 cache similar to the Optane nodes. There is a 30.25 MB L3 cache shared between all cores. Each socket has 6 DIMM slots each with a 32 GB DDR4 DRAM. That adds up to 384 GB of DRAM. Each node has 2 memory controllers connected to 6 DRAM DIMMs and all nodes are connected via single port to a Mellanox EDR 100 GB/s switch. We also utilize a 8 node DRAM-only setup with Skylake nodes with half the memory capacity.

We use a different architecture for DRAM nodes because of the lack of availability of Cascade-Lake based machines as they have only recently been introduced on the market. Cascade-Lake architecture is essentially the same as Skylake with a larger package size, 4% higher CPU frequency and slightly higher data rate [6]. Although the Optane nodes has a higher L3 cache size, the cache size/core ratio is equivalent for both architecture. The DRAM nodes have slightly slower cores but they are in higher number for 4 nodes in total compared to the Optane node. The Optane node consumes 25% more power than a single DRAM node due to additional DIMMS but has more than 4x memory capacity. The other heterogeneous memory architecture available today is HBM-DRAM hybrid memory systems like Intel KNL. We do not evaluate this architecture, as HBM is a very costly resource in terms of its smaller capacity and aquisitional cost. Our objective is to increase the memory-per-core ratio in HPC clusters while reducing cost and energy consumption to solve larger problem sizes. We used the LMbench 3.0 [18] benchmark to compare the memory bandwidth and access latency of both the Optane node and a DRAM node. We used a buffer size of 4 GB with a single instance of the benchmark and averaged all the readings over 3 runs. The results are presented in Table 3. The DRAM node has up to 2% better read bandwidth compared to Optane node but up to 5% lower write bandwidth. The same difference is reflected in the sequential and random access latencies as well. The acquisition cost of a single DRAM node is approximately 0.65 times the cost of a single Optane node as shown in Table 2. Hence, the total acquisition cost of 4 DRAM nodes is 2.5x the cost of a single Optane node, which provides an equivalent memory address space. These cost ratios are devised from the actual quotes of the nodes used for experimentation in our paper which cannot be revealed due to non-disclosure agreements. All the nodes are connected to rack mounted power distribution units (PDU) provided by Hewlett Packard Enterprise, which send Protocol Data Units to the administrator node. We capture the basic power data of the data units from the rack for DRAM nodes. These data units are timestamped data at the outlet granularity so we get per-node power data as well. We capture the same data for the Optane nodes.

Table 3: Performance comparison using lmbench3.0

Node	Rd BW(MB/s)	Wr BW(MB/s)	Sequential access(ns)	Random access(ns)
Optane	10926 ± 879	8792 ± 15	27.2 ± 0.1	85.4 ± 1.1
DRAM	11153 ± 712	8353 ± 15	26.8 ± 0.1	92.5 ± 1.1

5 EXPERIMENTS

We evaluate the performance of a HPC application and 2 HPC proxy-apps on the Optane nodes and DRAM nodes. Optane DC can be configured to run in different modes: it can be used as a byte-addressable memory with DRAM as a direct-mapped cache (Memory mode), or as a persistent memory device separate from the memory (App-direct mode), or as a combination of the two (Mixed mode). In order to have the applications allocate their heap data on Optane DC and DRAM agnostically, we modified two files in the Linux OS kernel, arch/x86/platform/efi/efi.c and arch/x86/boot/compressed/eboot.c, to treat NVDIMM as DRAM [13], which unifies the Optane DC and DRAM under one large address space in true DRAM-NVM hybrid memory fashion. We refer to this mode as the "Flat" mode hereafter.

We compare the performance of the single Optane node operating in Flat and Memory-mode separately with a cluster of 4 DRAM nodes with an equivalent amount of DDR4 DRAM main memory as described in Table 2. We also perform the evaluation of the same benchmarks running over 2 Optane nodes operating in Flat mode connected by a high speed interconnect. The aim of these experiments is to compare the performance of homogeneous address spaces connected over high-speed interconnects to massive address spaces with a hybrid but slower memory architecture with fewer nodes. We want to evaluate whether using NVM as a part of the primary memory architecture can deliver reasonable performance on fewer number of nodes compared to the conventional DRAM address spaces spread across a larger number of nodes. We scale the problem sizes of our benchmarks to ensure they do not fit entirely in the DRAM of a single Optane node so that we do not end up measuring only the performance of DRAM. However, in the multi-node Optane setup, smaller problem sizes do execute only on DRAM. Our problem sizes can be described as large HPC problems ranging from 200 GB to 1.3 TB. The memory footprint of each application is indicated in Table 4. The upper limit of the problem sizes were fixed such that the applications did not run into an out of memory error on any of the setups. The memory allocation for all our benchmarks is memory technology agnostic. In Flat mode, it exhausts local DRAM memory before spilling onto NVM. The amount of compute we scale up to is bound by the amount of compute available on the Optane node to have a fair comparison. We use hyper-threading on the single Optane node in order to scale the compute on the DRAM nodes to resemble a similar core-to-working-set-size ratio compared to bare DRAM nodes and the 2 Optane nodes.

Table 4: Benchmark configuration

Benchmark	No. of iterations	Memory Footprint	
		Strong Scaling(GB)	Weak Scaling(GB/process)
AMG	Variable(19-21)	388	9.6
VPIC	600 timesteps	572	12
LULESH	10	590	22

We perform strong and weak scaling of applications by using MPI on both Optane and DRAM nodes. We use Open MPI 3.1.3. We use LIKWID [32] to collect performance characteristics such as energy consumption and memory bandwidth on all nodes for all the experimental runs. All processes are pinned to a particular

core individually for every run on both setups. Hence, the memory mapping for every process is consistent across runs even if the processes mapped to DRAM will have different performance compared to processes mapped to NVM. We also calculate the total energy consumed by the nodes based on rack managed PDU data. The racks were provided by HP and utilize model H8B50A full-length PDUs. These PDUs do not collect the power consumption for the Mellanox switch and the cooling system used for the nodes. The PDUs report the power consumption of the node at a 10 second interval. We utilize "ipmitool" to collect node power numbers.All applications are compiled using GCC 7.3.0 on both setups with -O2 optimization. We do not use any special libraries that are not provided with the source code of these applications and the page size used on both setups is 4 KB. We next describe our applications used in experiments.

VPIC:. Vector Particle-In-Cell (VPIC) [5] models kinetic plasmas in 1 to 3 dimensions and employs a variety of short-vector, single-instruction-multiple-data (SIMD) intrinsics for high performance and has been designed so that the data structures align with cache boundaries making it compute bound. However, for our experiments we do not focus on vector operations. The code comprises of kernels that compute multiple data streams and operate on entire data structures. We use the 'lpi' input deck for our experiments.

AMG:. AMG is a parallel algebraic multi-grid solver for linear systems arising from problems on unstructured grids [33]. The driver provided with AMG builds linear systems for various 3-dimensional problems. It it is an SPMD code that uses MPI and OpenMP threading within MPI tasks. AMG is memory bound with only about 1-2 computations per memory access, so memory-access speeds will also have a large impact on performance. We use the default problem, which is a Laplace type problem on a cube with a 27-point stencil.

LULESH. LULESH [14] is a highly simplified application, hard-coded to only solve a simple Sedov blast problem with analytic answers. It features numerical algorithms, data motion, and programming style typical for scientific C or C++ based applications. It uses MPI and OpenMP for parallelization and is also memory bound.

These applications are representative of the workloads of common HPC applications that are currently used. Hence, evaluating the above applications will provide good approximation of the energy and cost efficiency of using a DRAM-NVM hybrid memory compared to a traditional memory architecture. We compare the inter-node and intra-node point-to-point MPI communication performance on DRAM and Optane, respectively, using the OSU MPI micro-benchmark suite 5.6.2 [7] as depicted in Figure 1. The y axes are on a logarithmic scale, with the left y-axis indicating bandwidth as a boxplot and the right y-axis showing latency as a line graph. We observe that the shared memory buffer communication on the Optane node has up to half the latency and 3x the bandwidth than the Infiniband-connected DRAM-only nodes. By using a Flat mode memory, we consolidate all MPI communication on the intra-node network (e.g., Intel Quickpath [35]), which can reduce the inter-node communication overhead for applications. This is also reflected in Table 5, where we present the MPI profiling information for all benchmarks evaluated in this work. We used EZTrace 1.1-9 [31] to collect the communication traces and observe that all applications spend less than 19% of their total execution time on MPI communication. The majority of the communication time is spent waiting (MPI_WaitAll) and during collective operations like MPI_AllReduce. As processes are spread over multiple nodes, there will be some node local and remote communication and the overall progress (e.g., timestep between stencil updates) would be upper bounded by the slowest link, i.e., any remote node point-to-point communication (for an MPI_WaitAll handle or collective). This will be true for any interconnect irrespective of the memory architecture on the nodes. However, the ratio of average communication time to total execution time for a given application remains constant for any number of nodes irrespective of the interconnect.

Table 5: MPI Communication profile (QP - Quickpath, IB - Infiniband)

Benchmark	MPI Processes	Comm. time(%)		Avg. msg size(MB)		Functions consuming most time	
		QP	IB	QP	IB	QP	IB
VPIC	48	4.73	2.8	0.38	0.31	MPI_WAIT (93%)	MPI_WAIT (53%)
AMG	48	18.8	27.38	0.022	0.025	MPI_WAITALL (72%)	MPI_WAITALL (85%)
LULESH	27	10.73	7.93	1.37	1.5	MPI_ALLREDUCE (65%)	MPI_ALLREDUCE (85%)

Due to the scale of the problem sizes, the execution times of the applications tends to be very high. In order to finish the experiments for VPIC and LULESH in a reasonable amount of time but also have a fair representation of the compute and memory operations, we reduce the number of iterations of the main loop. This reduction has no effect on the performance of the application. The total number of iterations each experiment runs is given in Table 4. We plot the average of execution times and energy consumption over three runs for each execution of the benchmarks with a standard deviation of 1 to 3% for execution time and 2 to 5% for energy. These applications are a fair representation of the HPC workloads that run on the current HPC machines.

6 RESULTS

Application bandwidth, energy consumption and execution time measurements are plotted in separate graphs to observe their correlation. These graphs are provided for both strong and weak scaling experiments. In all graphs, the values are indicated on the y-axis and depicted as bar charts.In the first set of graphs, Solve time is indicated on the left-hand y-axis with units as seconds(s) and depicted as a bar chart. In the second set of graphs, bandwidth is indicated on the left-hand side y-axis with units as megabytes/seconds (MB/s) and depicted as a bar chart. In the next set, energy is indicated on the left-hand side y-axis with units as Joules (J) and depicted as a bar chart. The axes for bandwidth and energy are on a logarithmic scale. The x-axis depicts the number of MPI processes for a given execution. We refer to the results of 4 nodes without NVM as "DRAM" and the two modes in which the single node with Optane DC PMMs operates by their respective names, "Flat-mode" and "Memory-mode". The 8 nodes with DRAM are referred to as "DRAM(8x)" and the 2 nodes with Flat-mode are referred to as "Flat multi-node".

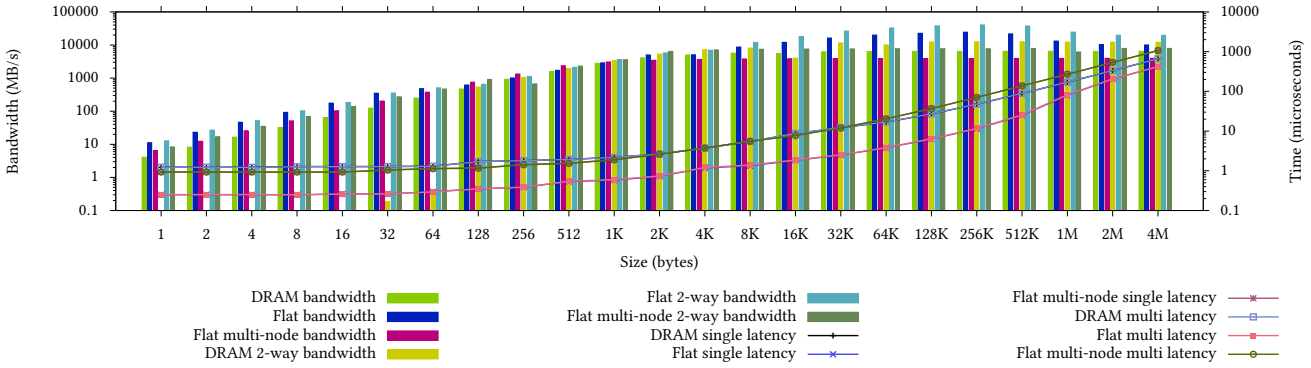
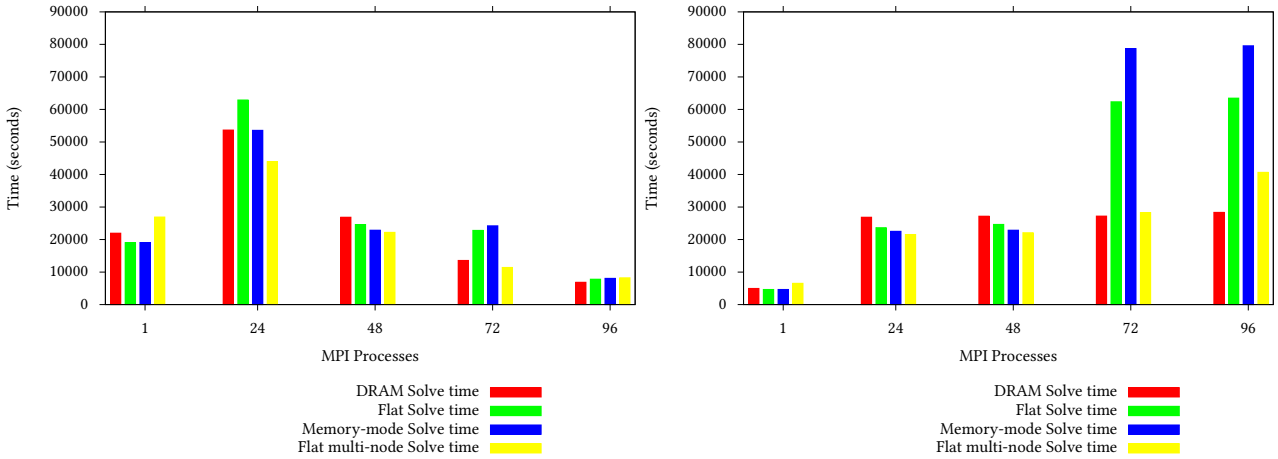


Figure 1: Performance of MPI communication for DRAM-only vs. Optane nodes



(a) VPIC Bandwidth Strong Scaling

(b) VPIC Bandwidth Weak Scaling

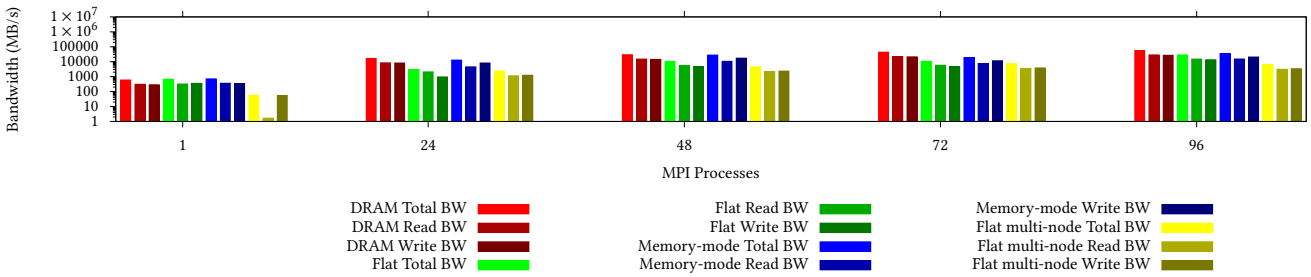
Figure 2: Solve time for VPIC

6.1 VPIC results

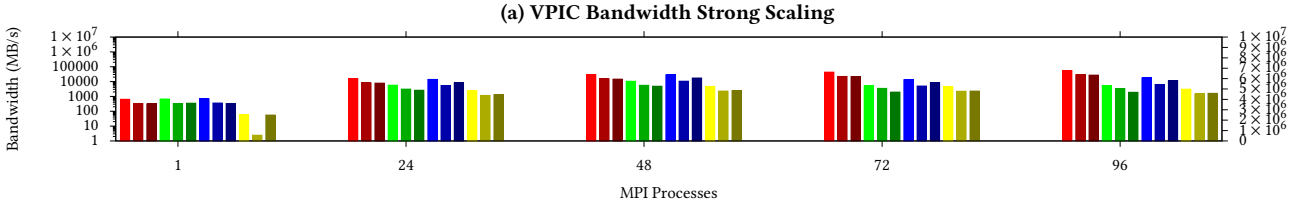
Figures 2a and 3a depict the execution times and application bandwidth for strong scaling of VPIC. We increase the number of processes from 1 to 96 and the overall problem size remains constant. To achieve this, we decrease the 'nppc' parameter from 131072 to 8192 in the input file.

Observation 1: *The execution times for Flat-mode, Memory-mode, Multi-node and DRAM are similar for VPIC with only up to 15% difference for all process counts up to the number of native cores (48), and time is further reduced by oversubscribing (hyper-threading beyond 48 cores). However, the bandwidth achieved by Flat-mode is lower than Memory-mode and DRAM barring the single process execution due to serialization. Nonetheless, Flat-mode delivers comparable performance to DRAM and Memory-mode provides a slight benefit over Flat-mode. We find a similar effect for execution time in weak scaling, except for oversubscriptions, which prolongs execution.*

This is due to the nature of VPIC to optimize cache hits (as mentioned in Section 5) and have strided access pattern. It is a compute-bound application that aligns data accesses with the cache line size. Due to its access pattern, VPIC obtains significant spatial locality in the cache and does not fetch memory frequently. This helps in hiding the memory access latency of NVM. VPIC also benefits from a larger L3 cache in the node with Optane DC and intra-node communication. Memory-mode executions try to hide latency for memory accesses by using the DRAM as a cache but page swapping between NVM and DRAM causes it to have only 15% decreased execution time compared to Flat-mode. This is due to the fact that the page swapping occurs every time a page fault occurs which is a very expensive operation in terms of memory access latency. The solve time is inclusive of the page swapping cost between DRAM and NVM. There is an increase in execution time under strong scaling at 24 processes for all executions relative to the serial execution for a single process. When multiple processes are executing at the same time, an overhead for parallelization is introduced first seen at 24 processes. As we increase the number of processes further, the benefit of parallelization outweighs the overhead. Memory mode has lower execution time than Flat Mode due to significantly higher write bandwidth achieved over Flat mode. We also see that the Flat-mode bandwidth increases at a faster rate than Memory-mode beyond 72 processes. This effect

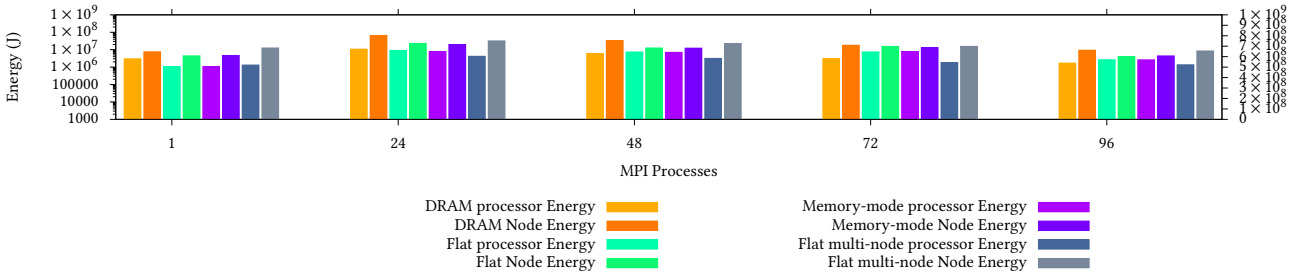


(a) VPIC Bandwidth Strong Scaling

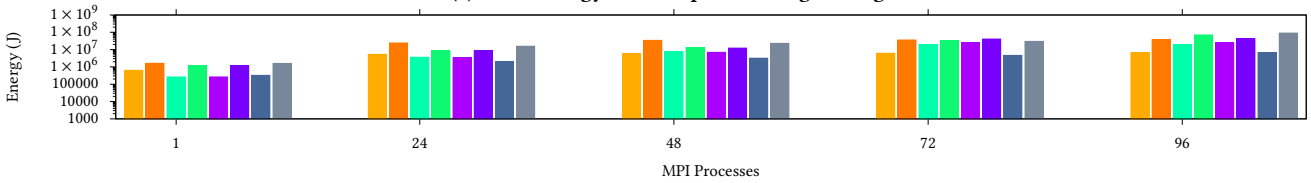


(b) VPIC Bandwidth Weak Scaling

Figure 3: Bandwidth measurement for VPIC



(a) VPIC Energy Consumption Strong Scaling



(b) VPIC Energy Consumption Weak Scaling

Figure 4: Energy consumption for VPIC

can be attributed to that fact that Flat-mode utilizes all 4 memory controllers at the same time whereas Memory mode utilizes only the 2 DRAM memory controllers. This can increase the load on the DRAM controllers with higher number of processes and restrict the bandwidth.

Figures 2b and 3b depict weak scaling, where the number of processes increases from 1 to 96 but the problem size per core remains constant. To achieve this, we keep nppc=32,768 in the input file for all runs. We observe that Flat-mode and Memory-mode have 10-17% lower execution time than DRAM up to 48 processes. The application bandwidth achieved by Flat-mode is lower than DRAM and Memory-mode executions. However, beyond 48 processes, the execution time doubles up due to oversubscribing of CPUs (hyper-threading). When oversubscribing, the memory controller queues overflow, which leads to serialization of loads and stores due to back pressure and nullifies the benefit of bank parallelism so that further scaling has no effect on the performance. This is also the reason for the longer times observed for 72 processes in strong scaling.

Figure 4a depicts the energy consumption for strong scaling of VPIC.

Observation 2: *Flat-mode, Multi-node and Memory-mode consume up to 3x less energy than DRAM for VPIC in terms of node energy due to similar execution times. These energy savings are observed for all*

executions in spite of similar or higher processor energy consumption for Flat and Memory-mode than DRAM.

The node energy is also contributed to by other auxiliary components, e.g., cooling fans and power supplies in addition to the processor and DIMMs. We measure the energy consumed by the processors separately using the RAPL driver through LIKWID but we do not measure the energy consumed by the auxiliary components. We account for that energy in the node energy measurement obtained from the PDUs. The node energy for Flat and Memory-mode and Multi-node remains lower due to the fewer number of nodes involved in the execution of the application, i.e., fewer components that consume energy. DRAM has a larger difference between its node energy and its processor energy due to more auxiliary components. The same effect is observed for weak scaling depicted in Figure 4b. In multi-node execution, as the number of nodes is twice than of Flat-mode, the energy consumed is also higher. The energy consumption of Flat-mode is up to 3x lower than DRAM for all executions barring the 96 processes case, where the higher execution time of Flat-mode causes it to consume more energy than DRAM. The higher execution time for Flat-mode is again a result of oversubscription. The Memory-mode consumes a similar amount of node energy compared to Flat-mode despite a slight improvement in execution time due to increase in energy consumption of DIMMS where DRAM is used as cache. The Flat multi-node execution has higher energy consumption than DRAM due to its higher execution time but increasing problem size the energy benefits should scale with capacity: With a problem size at the capacity of the 2 nodes with Flat memory, 8 DRAM-only nodes are required to fit the problem size in memory, and the energy consumption of the DRAM-only nodes is higher than the 2 nodes with Flat memory when extrapolated.

Inference 1: *Applications that optimize their cache hits (VPIC) or are compute-bound can benefit from a low power, high capacity memory device in terms of energy given a large enough last-level cache and enough compute resources on a single node. Their execution will result in low energy consumption under minimal to no performance degradation for a DRAM-NVM hybrid memory platform with lower acquisition and operation costs than multiple nodes using traditional DRAM memory. Exploiting DRAM as a cache for a NVM based memory can provide a slight advantage over the DRAM-NVM hybrid memory space in terms of execution time at the cost of a 20% reduction in problem size.*

6.2 AMG results

Figures 5a and 6a depict the results for strong scaling of AMG. We scale the number of processes from 24 to 96 and keep the aggregate problem size constant. This is done by using the input value of 768 in the x,y and z problem dimensions for a single process and reducing it to 192, 192 and 128 in x,y and z problem dimensions. We do not provide results for a single process execution as the problem size would exceed the memory address space of a single DRAM node. Weak scaling is depicted in Figures 5b and 6b, where we scale up the number of processes from 1 to 96 keeping the size per processor constant. This is achieved by keeping the input value of all dimensions at 224 for every run.

Observation 3: *The execution time of Flat-mode is an order of magnitude slower than DRAM for AMG. The total bandwidth achieved by Flat-mode is two orders of magnitude lower than DRAM, which causes the execution time to be an order of magnitude higher for both strong and weak scaling. Memory-mode provides up to a 20% reduction in execution time compared to Flat-mode for strong scaling with a higher memory bandwidth due to using DRAM as a cache. For weak scaling, Memory-mode has a 30% lower execution time compared to Flat-mode before core oversubscription and a 60% reduction in execution time after oversubscription.*

AMG is a memory-bound benchmark and its performance is heavily dependent on the write latency of the memory device. The access pattern is very irregular, where array indices are often indirectly referenced from other arrays. Due to higher write latency of Optane DC, Flat and Memory-mode bandwidth remains lower than DRAM and the execution time is higher. The gap between execution time widens as we scale up the number of processes for weak scaling. As the process count exceeds 48 cores for Flat-mode, the execution time increases dramatically due to oversubscription of resources. Memory-mode hides memory access latency by using DRAM as a cache but due to constant swapping of pages between DRAM and NVM the execution time is 6 to 9x higher than DRAM for strong scaling. In weak scaling, for lower number of processes the problem size is also lower, which results fewer page swaps between DRAM and NVM. Hence, the execution time is 30% lower compared to Flat-mode but 6 to 15x slower than DRAM. Figure 7a and Figure 7b depict the energy consumption of Flat-mode, Memory-mode and DRAM executions for strong and weak scaling of AMG, respectively.

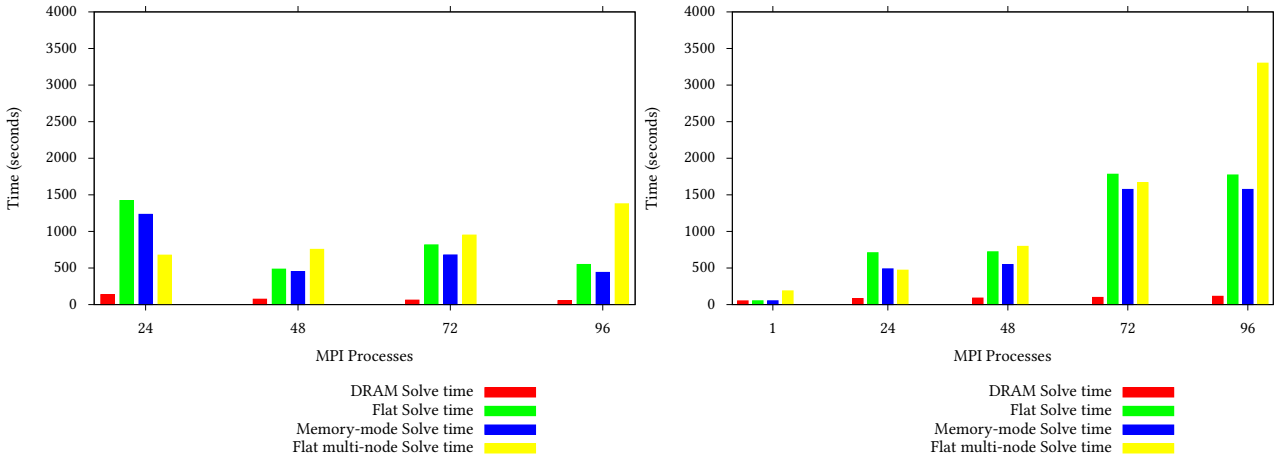
Observation 4: *For strong scaling of AMG, the total energy consumption of Flat-mode is up to 2x higher than the total energy consumption of DRAM. For weak scaling, the energy consumption for a single process execution in Flat and Memory-mode is lower than DRAM. In contrast, for all multi-process executions the energy consumption of Flat and Memory-mode is 3-4x higher than the energy consumption of DRAM.*

Energy consumption of an application is directly correlated to its execution time. Even though Optane DC DIMMs consume less power than DRAM DIMMs, they consume higher energy due to longer execution times. The DRAM DIMMs plus their additional processors consume a fraction of the aggregate energy of all 4 DRAM nodes. In contrast, the Optane DC DIMMs consume a large fraction of the total energy consumption of the Optane node. Memory-mode uses DRAM as cache causing the DIMM energy consumption to rise such that only a 12-15% reduction in node energy is observed compared to Flat-mode.

Inference 2: *Memory-bound applications like AMG, which are dependent on fast access latency of memory devices, will suffer in performance and energy when executed on a DRAM-NVM hybrid memory architecture. Using DRAM as a cache for NVM can help reduce execution time and energy consumption up to 15-20% at the cost of a 20% reduction in problem size. However, the acquisition costs of the multiple DRAM nodes required to run larger problems is more than 2x higher than a single Optane node.*

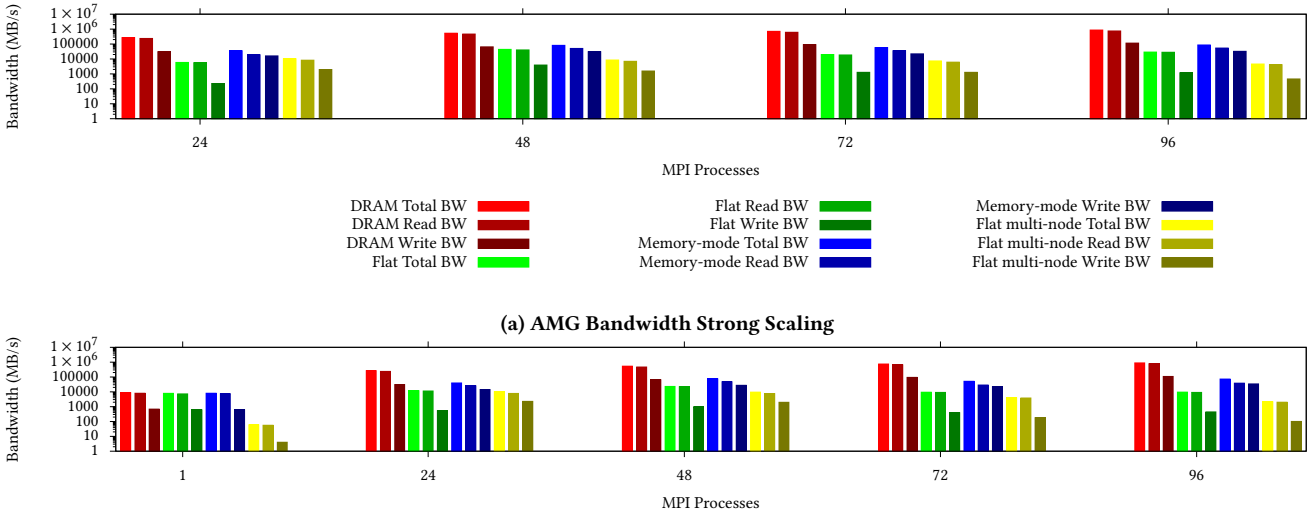
6.3 LULESH results

Figures 9a and 8a depict the bandwidth and execution time for strong scaling of LULESH from 8 to 64 MPI processes. We do not provide results for the single process execution as the problem



(a) AMG Solve time Strong Scaling **(b) AMG Solve time Weak Scaling**

Figure 5: Solve time for AMG



(a) AMG Bandwidth Strong Scaling

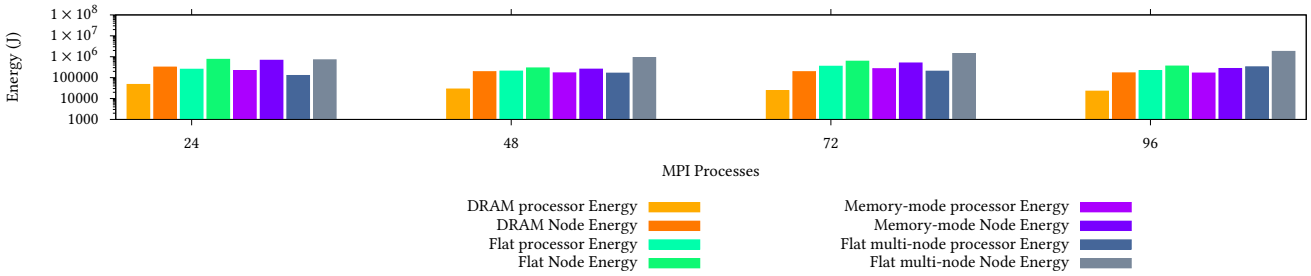
(b) AMG Bandwidth Weak Scaling

Figure 6: Bandwidth measurement for AMG

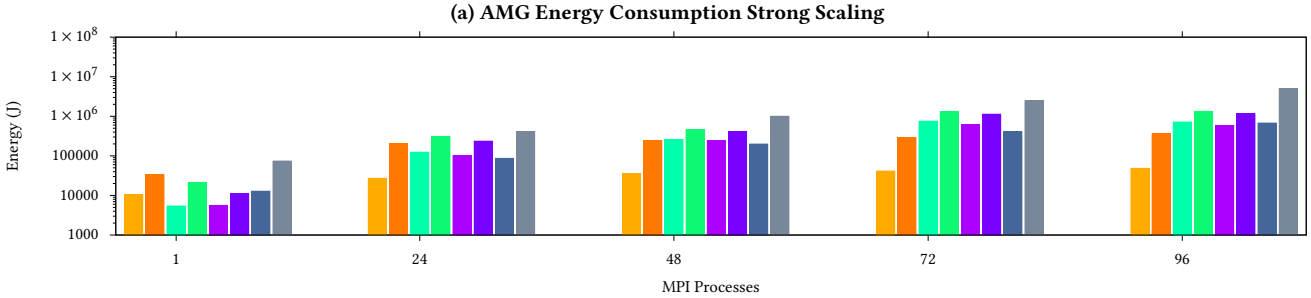
size exceeds the memory capacity of a single DRAM node. The aggregate problem size remains the same. In order to achieve this, the input parameter changes from 1024 to 256 in each dimension keeping the number of elements constant. Figure 9b and 8b depict the bandwidth and execution time for weak scaling, i.e., scaling the number of processes from 1 to 64 while keeping the problem size per processor constant. This is achieved by keeping the input parameter at 320 in every dimension, which results in increasing numbers of elements with the number of processes.

Observation 5: *For LULESH, the execution time for Flat-mode is up to an order of magnitude higher than DRAM; and Memory-mode is more than 50% faster than Flat-mode but 3-7x slower than DRAM. Due to the higher access latencies of Optane DC, Flat and Memory-mode do not achieve reasonable read or write bandwidth. DRAM, on the other hand, achieves high read and write bandwidth, which results in lower execution times. This effect is observed for both strong and weak scaling. For weak scaling, the execution time of Flat-mode is up to an order of magnitude higher than DRAM while Memory-mode is more than 60% faster than Flat-mode for larger numbers of processes.*

LULESH, like AMG, is also heavily dependent on access latency on the underlying memory device making it a memory-bound application. The memory access patterns of LULESH are non-unit stride due to its region based solvers [14]. This increases the LLC misses and wastes precious cycles waiting for memory fetches as observed in [23]. For weak scaling, fewer processes on the Optane node for Flat and Memory-mode achieve comparable bandwidth to DRAM. Hence, the difference in execution time is not significant. However, as we scale up the processes beyond 27, the execution time increases

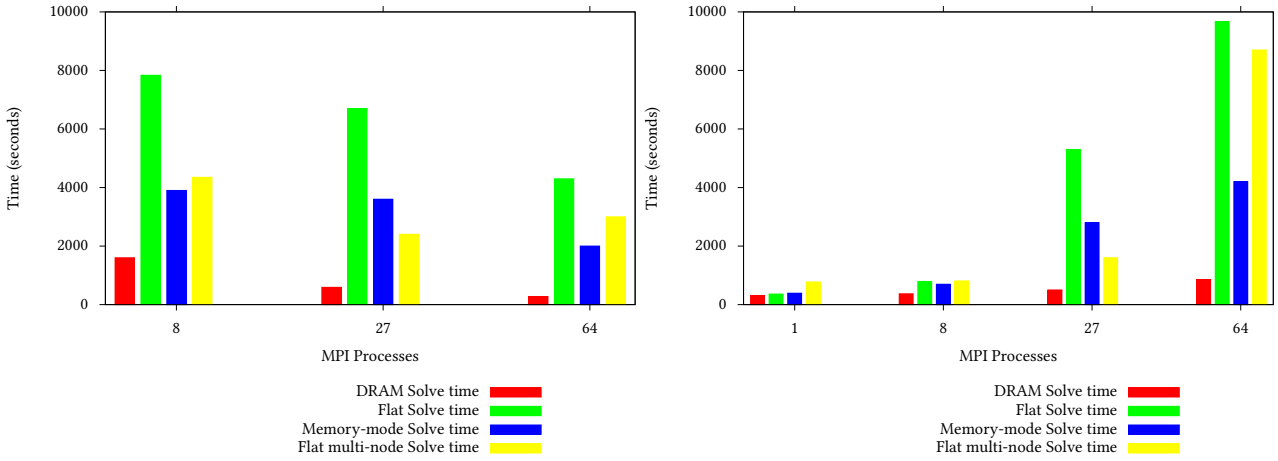


(a) AMG Energy Consumption Strong Scaling



(b) AMG Energy Consumption Weak Scaling

Figure 7: Energy consumption for AMG



(a) Strong Scaling **(b) Weak Scaling**

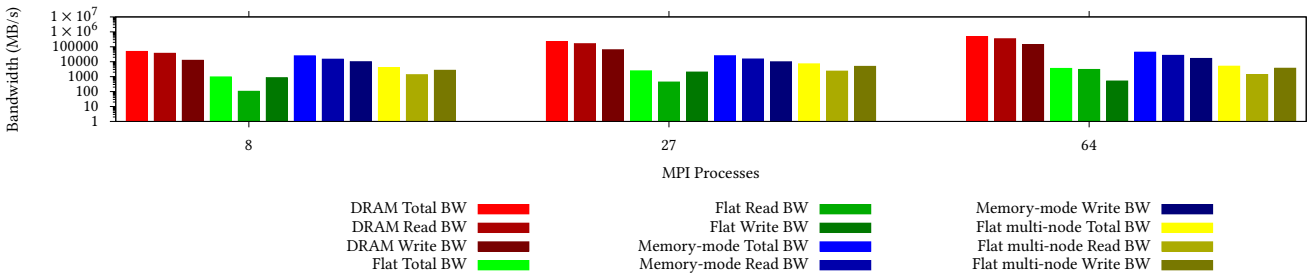
Figure 8: Solve time for LULESH

exponentially due to low bandwidth. The application bandwidth for LULESH is also affected by problem size. In strong scaling, as the problem size per process reduces, the memory bandwidth increases with higher number of processes. However, in weak scaling where the problem sizes per process remains the same, the application memory bandwidth reduces. This could be a result of the non-unit stride references of LULESH. Memory-mode executions perform significantly better than Flat-mode due to DRAM caching but are still 3-7x slower than DRAM. This is due to an order of magnitude higher bandwidth than Flat mode execution.

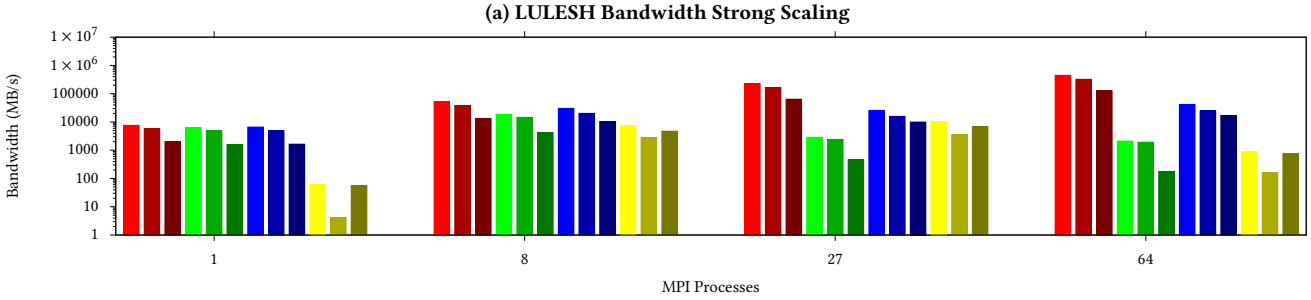
Figures 10a and 10b depict energy consumption and execution times for strong and weak scaling of LULESH respectively.

Observation 6: *LULESH consumes up to 3x more energy on Flat-mode than on DRAM due to longer execution time for strong scaling. For weak scaling, Flat and Memory-mode consumes 30% less energy than DRAM up to 8 processes.*

The Optane DIMMs contribute heavily to the total energy consumption of Flat and Memory-mode executions compared to DRAM energy consumption on DRAM nodes in strong scaling. Compared to Flat-mode, Memory-mode consumes up to 50% less node energy due to the significantly lower execution times. In weak scaling, due

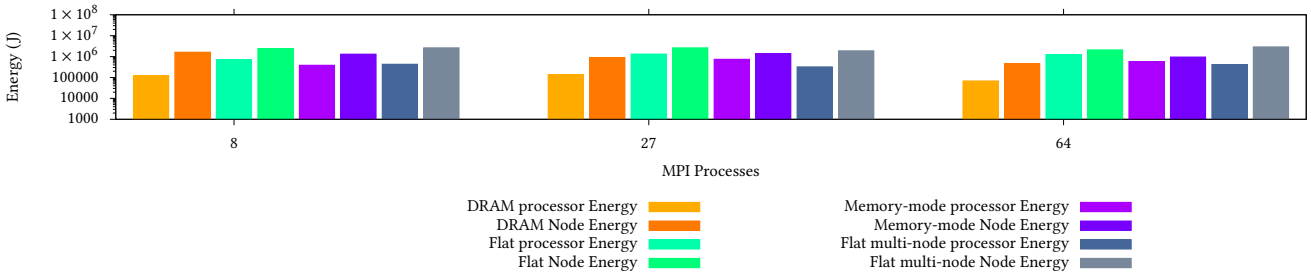


(a) LULESH Bandwidth Strong Scaling

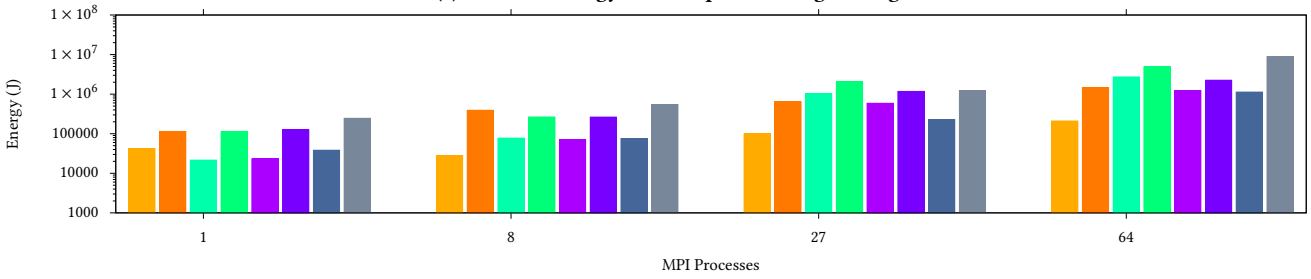


(b) LULESH Bandwidth Weak Scaling

Figure 9: Bandwidth measurement for LULESH



(a) LULESH Energy Consumption Strong Scaling



(b) LULESH Energy Consumption Weak Scaling

Figure 10: Energy consumption for LULESH

to lower power consumption of Optane DIMMs and fewer processors with similar execution times, Flat-mode consumes less energy. The energy consumption increases drastically for Flat-mode beyond 8 processes due to higher execution times. However, Memory-mode executions have similar energy consumption to DRAM for higher number of processes. These results show that applications dependent on both read and write bandwidth for performance can expect reasonable performance with some energy savings for smaller problem sizes and fewer processes on a single node with Optane DC

combined with lower acquisition costs. Using Memory-mode, we can obtain significant performance benefits compared to Flat-mode.

Inference 3: *From the above observations, we infer that compute-bound applications, which utilize cache locality well, are able to run larger problem sizes on fewer compute nodes with a DRAM-NVM hybrid memory system. They experience minimal performance degradation while providing significant cost and energy savings. Conversely, memory-bound applications suffer from performance degradation and higher energy cost. Using DRAM as a cache for NVM can reduce the performance degradation for memory-bound applications. There may also be an opportunity for memory-bound applications on Flat-mode if memory latency can be hidden using aggressive prefetching. Hence, a single node with a DRAM-NVM hybrid memory system can support large problem sizes with reasonable trade-offs that traditionally would require approximately 4 DRAM-only nodes to run.*

Observation 7: *The executions with split allocations utilizing more DRAM have lower execution times and consequently lower energy consumption.*

Figure 11 depicts execution time and energy consumption for all three benchmarks while splitting the dynamic memory across DRAM and NVM in fixed ratios of 1:1, 1:3 and 1:7 (DRAM:NVM). We achieve this by overloading the allocation wrappers in the benchmarks and utilizing 'numa_alloc_onnode()' instead of 'malloc()'. The problem sizes used for these experiments are smaller than the previous experiments as we are limited by DRAM capacity. The left-hand y-axis depicts energy consumption on a logarithmic scale and the right-hand y-axis depicts time in seconds on a linear scale except in Figures 11e and 11f. For VPIC, we observe that DRAM(8x) consumes more energy than all other executions despite having lower execution time until the other executions start using hyper-threading. This can again be attributed to the cache locality of VPIC and fewer nodes to execute on due to the consolidated memory capacity of Flat-mode. For AMG, any execution that utilizes NVM has higher execution times and the ratio of DRAM memory used has no effect. Hence, the energy consumption for all Flat-mode executions is higher than DRAM(8x). For LULESH, we see that DRAM(8x) has lower execution times and energy consumption compared to other executions but splitting memory across DRAM and NVM mitigates the performance degradation.

Inference 4: *From the above observations, we infer that applications with cache locality benefit from a memory allocations split across DRAM and NVM. Higher utilization of DRAM can mitigate the performance degradation due to NVM.*

6.4 Projection

To assess the cost benefit of a DRAM-NVM memory system based supercomputer, we refer to Frontera, a cluster with Cascade Lake processors, predecessors to the Sapphire Rapids processors to be used in Aurora [3]. It has an acquisition cost of $60 million [30] and achieves 24 petaPFLOPS of LINPACK performance on 8,008 nodes connected by Mellanox HDR-100 switches with dual sockets and 192 GB of DRAM on each node that amounts to 1.46 PB in total. Each node costs approximately $7,500 and we spread the cost across Compute, Memory and Interconnect with ratios of 0.66 ($5,000), 0.2 ($1,500) and 0.14 ($1,000), respectively. We assume these ratios based on an informal survey of costs for the respective components that is available publicly as the exact quotes for the system are unavailable.

The Cascade Lake nodes can support up to 768 GB of DRAM using the 32 GB DDR4 DIMMs and 6 TB of Optane DC using the 512 GB NVDIMMs based on memory channels shown in Table 2. This brings the support to a total of approximately 55 PB of byte-addressable memory in Flat mode. Based on the cost ratio for Optane DC compared to DRAM in Section 1, Optane DC on a single node costs around $25,000 and the extra DRAM adds another $4,500, which increases the per-node cost to $37,000. Just the addition of Optane DC PMMs and the required DRAM to support it will bring the cost of Frontera up to $296 million. If compared to the projected acquisition cost ($500 million) of Aurora [3], we still would have around $200 million to purchase additional compute to increase additional FLOPS for this machine, which is equivalent to the entire budget of Summit [29]. However, if the current DRAM based architecture of Frontera was to be scaled up to support a problem size of 55 PB, it would require 290,290 nodes and would cost $2.2 billion. Even though the compute performance of such a machine will be approximately 708 petaFLOPS assuming a linear speed up, the acquisition cost and power required to operate such a large cluster makes it infeasible. We can bring down the number of nodes by upgrading the amount of DRAM in every node to 768 GB. Such a machine requires approximately 72,573 nodes and the cost of the cluster will be approximately $871 million, where each node costs $13,000. This configuration will achieve a compute performance of approximately 177 petaFLOPS assuming a linear speed up, but it will come at approximately 1.25x the cost of Aurora, and its power and space requirements will still be infeasible.

By just increasing the capacity of the current cluster by adding NVM, one can support 20x larger problem sizes than Summit at just 1.5x the cost. As seen in Fig. 4b, compute-bound application will suffer minimal performance degradation while increasing the problem size support by approximately 37x and being significantly more energy efficient than any DRAM based configurations. Memory-bound applications will suffer while using a memory agnostic allocation scheme as seen in Fig. 6b. One can sacrifice 11% of the total memory capacity by using DRAM as cache for NVM to limit the slowdown up to 50% for some applications while still supporting 19x larger problem sizes than Summit [23]. Also, a memory aware allocation scheme can limit the slowdown by keeping memory bound data structures in DRAM [25] for a hybrid memory system.

7 CONCLUSION

Our work assessed the performance and energy characteristics of DRAM-NVM memory systems for large problem sizes under different modes and compared it to DRAM-only memory systems. For memory-bound applications, using a DRAM-NVM memory can hamper performance due to higher access latencies but using DRAM as a cache for NVM can restrict the performance degradation to a certain extent with the tradeoff in smaller problem sizes. However, with enough compute resources, compute-bound applications achieve similar performance and lower energy on Flat-mode compared to traditional HPC systems. Flat-mode has marginally lower performance than Memory-mode for most cases. But Flat-mode allows significantly larger problems to be brought into memory, a

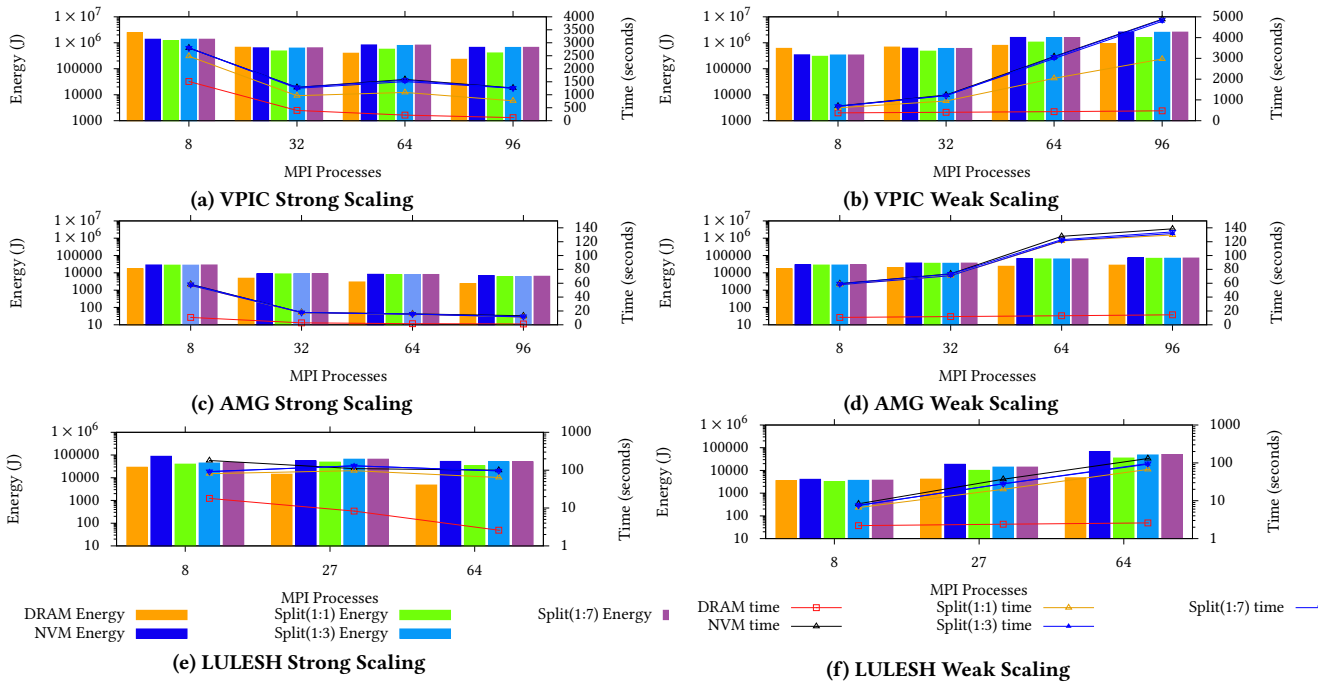


Figure 11: Energy consumption of VPIC, AMG and LULESH under different memory allocation policies

tradeoff future HPC system designers will have to consider when choosing between fast DRAM or slower but larger NVM at equal acquisition cost. Memory allocation policies that can utilize DRAM more can also help take advantage of NVM. Using NVM to extend byte-addressable memory in HPC clusters with or without DRAM as a last-level cache provides a viable option to reduce the gap between compute and memory scaling. It provides a promising path to extend memory address spaces of HPC systems without compromising performance and energy for certain codes, which translates into lower operational cost combined with lower acquisition costs.

ACKNOWLEDGMENTS

This material is based upon work supported by United States Department of Energy National Nuclear Security Administration prime contract #89233218CNA000001 subcontract #508854 dated November 1, 2018 for Los Alamos National Laboratory, NSF grant 1525609 and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. The unclassified release number is LA-UR-19-27887.

REFERENCES

[1] Dmytro Apalkov, Alexey Khvalkovskiy, Steven Watts, Vladimir Nikitin, Xueti Tang, Daniel Lottis, Kiseok Moon, Xiao Luo, Eugene Chen, Adrian Ong, Alexander Driskill-Smith, and Mohamad Krounbi. 2013. Spin-transfer Torque Magnetic Random Access Memory (STT-MRAM). *J. Emerg. Technol. Comput. Syst.* 9, 2, Article 13 (May 2013), 35 pages. https://doi.org/10.1145/2463585.2463589

[2] Muhammad Usman Ashraf, Amna Arshad, and Rabia Aslam. 2019. IMPROVING PERFORMANCE IN HPC SYSTEM UNDER POWER CONSUMPTIONS LIMITATIONS. *International Journal of Advanced Research in Computer Science* 10, 2 (2019).

[3] Aurora - May 2019 [n. d.]. Aurora - May 2019. https://press3.mcs.anl.gov/aurora/.

[4] Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Kerry Hill, Jon Hiller, et al. 2008. Exascale computing study: Technology challenges in achieving exascale systems. *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep* 15 (2008).

[5] K. J. Bowers, B. J. Albright, L. Yin, B. Bergen, and T. J. T. Kwan. 2008. Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation. *Physics of Plasmas* 15, 5 (2008), 055703. https://doi.org/10.1063/1.2840133 arXiv:https://doi.org/10.1063/1.2840133

[6] cascadelake [n. d.]. Cascade Lake - Microarchitectures - Intel, WikiChip 2019. https://en.wikichip.org/wiki/intel/microarchitectures/cascade_lake#Memory_Hierarchy.

[7] D. K. Panda et al. [n. d.]. OSU Micro-Benchmarks. https://mvapich.cse.ohio-state.edu/benchmarks/.

[8] Frontier - Nov 2019 [n. d.]. Frontier - Nov 2019. https://www.cray.com/company/news-and-media/doe-frontier-press-release.

[9] Abdoulaye Gamatié, Alejandro Nocua, Joel Weloli, Gilles Sassatelli, Lionel Torres, David Novo, and Michel Robert. 2019. Emerging NVM Technologies in Main Memory for Energy-Efficient HPC: an Empirical Study. (2019).

[10] Abhishek Gupta and Dejan Milojicic. 2011. Evaluation of hpc applications on cloud. In *2011 Sixth Open Cirrus Summit*. IEEE, 22–26.

[11] Jonathan Hines. 2018. Stepping up to Summit. *Computing in science & engineering*. 20, 2 (2018), 78–82.

[12] HPC evolution [n. d.]. HPC evolution. https://insidehpc.com/2016/08/the-evolution-of-hpc/.

[13] Joseph Izraelevitz, Jian Yang, Lu Zhang, Juno Kim, Xiao Liu, Amirsaman Memaripour, Yun Joon Soh, Zixuan Wang, Yi Xu, Subramanya R. Dulloor, Jishen Zhao, and Steven Swanson. 2019. Basic Performance Measurements of the Intel Optane DC Persistent Memory Module. *CoRR* abs/1903.05714 (2019). arXiv:1903.05714 http://arxiv.org/abs/1903.05714

[14] Ian Karlin, Jeff Keasler, and Rob Neely. 2013. *LULESH 2.0 Updates and Changes*. Technical Report LLNL-TR-641973. 1–9 pages.

[15] Benjamin C Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2009. Architecting phase change memory as a scalable dram alternative. *ACM SIGARCH Computer Architecture News* 37, 3 (2009), 2–13.

[16] Bao Li and Pingjing Lu. 2015. The Evolution of Supercomputer Architecture: A Historical Perspective. In *CCF Conference on Computer Engineering and Technology*. Springer, 145–153.

[17] Teng Ma, Mingxing Zhang, Kang Chen, Zhuo Song, Yongwei Wu, and Xuehai Qian. 2020. AsymNVM: An Efficient Framework for Implementing Persistent Data Structures on Asymmetric NVM Architecture. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages*

and Operating Systems (ASPLOS '20). Association for Computing Machinery, New York, NY, USA, 757–773. https://doi.org/10.1145/3373376.3378511

[18] Larry W McVoy, Carl Staelin, et al. 1996. lmbench: Portable tools for performance analysis.. In *USENIX annual technical conference*. San Diego, CA, USA, 279–294.

[19] Onur Mutlu. 2013. Memory scaling: A systems architecture perspective. In *2013 5th IEEE International Memory Workshop*. IEEE, 21–25.

[20] Ravi Nair. 2015. Evolution of memory architecture. *Proc. IEEE* 103, 8 (2015), 1331–1345.

[21] Objective analysis [n. d.]. PCM become a reality. http://www.objective-analysis.com/uploads/2009-08-03_Objective_Analysis_PCM_White_Paper.pdf.

[22] M. Ben Olsen, Brandon Kammerdiener, Michael R. Jantz, Kshitij A. Doshi, and Terry Jones. 2019. Portable Application Guidance for Complex Memory Systems. In *Proceedings of the fifth ACM/IEEE International Symposium on Memory Systems*. ACM/IEEE, 156–166.

[23] Onkar Patil, Latchesar Ionkov, Jason Lee, Frank Mueller, and Michael Lang. 2019. Performance characterization of a DRAM-NVM hybrid memory architecture for HPC applications using Intel Optane DC Persistent Memory Modules. In *Proceedings of the fifth ACM/IEEE International Symposium on Memory Systems*. ACM/IEEE, 288–303.

[24] Ivy Peng, Kai Wu, Jie Ren, Dong Li, and Maya Gokhale. 2020. Demystifying the performance of HPC scientific applications on nvm-based memory systems. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 916–925.

[25] Ivy B. Peng, Maya B. Gokhale, and Eric W. Green. 2019. System evaluation of the Intel Optane Byte-addressable NVM. In *Proceedings of the fifth ACM/IEEE International Symposium on Memory Systems*. ACM/IEEE, 304–315.

[26] T Peterka, K Moreland, K Isaacs, B Muite, D Pleiter, B Raffin, U Rüde, R Sisneros, and GH Weber. 2019. 4.4 Exascale Systems. *In Situ Visualization for Computational Science* (2019), 18.

[27] Judy Qiu, Shantenu Jha, Andre Luckow, and Geoffrey C Fox. 2014. Towards HPC-ABDS: an initial high-performance big data stack. *Building Robust Big Data Ecosystem ISO/IEC JTC* 1 (2014), 18–21.

[28] Roberto Rodríguez-Rodríguez, Fernando Castro, Daniel Chaver, Rekai González-Alberquilla, Luis Piñuel, and Francisco Tirado. 2014. Write-aware replacement policies for pcm-based systems. *Comput. J.* 58, 9 (2014), 2000–2025.

[29] Summit [n. d.]. SUMMIT. https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/.

[30] TOP500 List - Nov 2020 [n. d.]. TOP500 List - Nov 2020. https://www.top500.org/lists/top500/2020/11/.

[31] François Trahay, François Rue, Mathieu Faverge, Yutaka Ishikawa, Raymond Namyst, and Jack Dongarra. 2011. EZTrace: a generic framework for performance analysis. In *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 618–619.

[32] Jan Treibig, Georg Hager, and Gerhard Wellein. 2010. Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In *2010 39th International Conference on Parallel Processing Workshops*. IEEE, 207–216.

[33] Ulrike Meier Yang et al. 2002. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics* 41, 1 (2002), 155–177.

[34] Fang Zhou, Song Wu, Youchuang Jia, Xiang Gao, Hai Jin, Xiaofei Liao, and Pingpeng Yuan. 2019. VAIL: A Victim-Aware Cache Policy to Improve NVM Lifetime for Hybrid Memory System. *Parallel Comput.* (2019).

[35] Dimitrios Ziakas, Allen Baum, Robert A Maddox, and Robert J Safranek. 2010. Intel® quickpath interconnect architectural features supporting scalable system architectures. In *2010 18th IEEE Symposium on High Performance Interconnects*. IEEE, 1–6.

[36] Darko Zivanovic, Milan Radulovic, Germán Llort, David Zaragoza, Janko Strassburg, Paul M Carpenter, Petar Radojković, and Eduard Ayguadé. 2016. Large-memory nodes for energy efficient high-performance computing. In *Proceedings of the Second International Symposium on Memory Systems*. 3–9.