Evaluation of Non-Volatile Memory Based Last Level Cache Given Modern Use Case Behavior

Alexander Hankin, Tomer Shapira, Karthik Sangaiah*, Michael Lui*, Mark Hempstead

Tufts University, *Department of Electrical and Computer Engineering*
*Drexel University, *Department of Electrical and Computer Engineering*
{hankin, tshapi02, mark}@ece.tufts.edu, {ks499, mdl45}@drexel.edu

***Abstract*—To confront the memory wall and keep up with the demands of changing use cases, Non-Volatile Memories (NVMs) have begun to be considered as a replacement for SRAM in the Last Level Cache (LLC). Recent work has shown that the small cell size of NVMs like *Spin- Torque Transfer RAM* (STTRAM) and *Resistive RAM* (RRAM) allows designers to build significantly denser LLCs than those with SRAM-based cells. In some cases, this allows for storing up to $10\times$ more data on-chip than before. As the working set size of use cases increases with the advent of statistical inference (e.g., machine learning (ML) and artificial intelligence (AI)), more capacity close to the processor is necessary to keep up with the demand for performance and low power.**

Despite the growing potential of NVM-based LLCs, there are still fundamental problems that need to be addressed. First, the research community is lacking a methodology for consistently modeling these devices, which leads to apples-to-oranges comparisons across NVM-based LLCs. Second, NVMs exhibit a key operational difference with SRAM: read and write asymmetry. The effects of this asymmetry on use case performance and power are mostly unknown with prior art relying only on total read and write counts and on limited sets of use cases.

In this work we present two novel contributions: (1) a set of heuristics for modeling emerging NVM-based LLCs, and (2) a workload characterization framework that learns how architecture-agnostic features, like entropy and working set size, affect the performance and power of a NVM-based LLC system for different use cases. In addition, with this work we release our NVM cell models and make them publicly available online. Using our NVM-based LLC models we show that NVM-based LLC energy use is up to an order of magnitude less than that of an SRAM-based LLC while ED^2P is generally on par. From our workload characterization framework, we show that for the AI use cases, energy and speedup are 99% correlated with write entropy, 90% write footprint, and unique write footprint while negligibly correlated with total read and write footprint.

I. Introduction

Memory system paradigm shifts are beginning to occur as a result of the memory wall and changing use case behavior. The working set size of applications is increasing, which puts increasing pressure on the memory to quickly supply large quantities of data. As the LLC is the processors last line of defense against an expensive off-chip memory access, a significant burden has fallen upon it. Traditional SRAM-based LLCs are density-limited and cannot scale to accommodate the growing working set sizes of ML and AI use cases. Improving the management of existing SRAM capacity is a very active area of research [1]–[5]; however, there is only so much that can be done. This has led the research community to explore the impact of NVM-based LLCs.

Beginning decades ago as a storage solution, NVMs have slowly made their way down the memory hierarchy, as the technologies have matured and the demand for more memory capacity has increased. Current emerging NVMs now not only offer high density and low leakage, but also fast read latencies which make them a competitor to modern SRAM technologies for the LLC [6]–[9]. Despite this potential, there are a few considerable roadblocks in the way of complete adoption of NVMs in the LLC including write performance and lifetime [7]–[13].

Prior art in NVM-based LLCs is recent and has primarily been focused on mitigating the issues with write performance, write energy and reliability [7]–[9], [14]–[19]. Existing work can be categorized into three groups:

1) Existing architectural techniques adapted for NVMs, e.g., wear leveling [20]
2) Novel architectural techniques, e.g., cache bypassing [21], [14], [16], [17]
3) Device level techniques, e.g., trading off different device-level parameters [15], [18], [22], [23]

To facilitate apples-to-apples comparisons across NVM-based LLCs in the research community, we present a set of modeling heuristics. With any emerging technology, there is a challenge in deriving the accurate software models needed for fast-paced simulation-based research. The lack of a consistent modeling methodology for NVM-based LLCs leads to apples-to-oranges comparisons in different works. Thus, our first contribution is to outline a unified modeling framework which researchers can use to compare NVMs.

Our second contribution focuses on understanding the effects of the unique operating behavior of NVMs in the context of the last level cache. Particularly, we study the extent to which use case memory behavior exacerbates or enhances system performance and energy efficiency for different NVM-based LLCs. This is because NVMs have unique characteristics, such as read and write asymmetry, which may better serve certain use cases over others depending on their behavior. In fact, there is variation in operating behavior even across individual technologies within the same class (where class is STTRAM or PCRAM, for example). Therefore, we explore the extent to which architecture-agnostic features affect system

performance and energy. This will allow a designer to select the correct NVM technology for her or his system, depending on target use case(s). We study this relationship across multiple Spin-Torque Transfer RAM (STTRAM) and Resistive RAM (RRAM) based LLCs.

For our workload characterization of memory behavior, we carefully select workloads which represent a wide range of use cases. We select traditional serial use cases as well as use cases which reflect modern and future applications, such as the highly parallelized, artificial intelligence (AI) applications. We characterize these workloads using a rigorous collection of memory behavior metrics [24] including address entropy, spatial locality, unique address footprint, and working set size. We analyze the variation in performance and energy across workloads and determine the amount of correlation with the architecture-agnostic features. We then discuss the extent to which the best LLC technology depends on memory behavior of the use case.

The structure of the rest of the paper is as follows: Section II gives relevant and necessary background information on the NVMs studied in this work; Section III presents our heuristics for NVM-based LLC modeling; Section IV outlines our simulation methodology; Section V presents the simulations results as well as our analysis; Section VI examines the relationship between architecture-agnostic workload behavior and NVM-based LLC; Section VII outlines next steps for our work; and we conclude with Section VIII, which offers our final thoughts.

II. Background

The most prominent emerging NVMs in recent years include PCRAM, STTRAM, and RRAM. More technologies continue to emerge including Charge Trap Transfer RAM (CTTRAM) as well as technologies which can leverage multiple bits per single level cell. In addition, the VLSI and architecture research communities continue to produce novel device-level and structural innovations, like multi-level cell (MLC) NVMs and 3D chips [10]. Table I below provides a brief summary of the technology classes used in this work.

	Advantages	**Drawbacks**
Phase Change RAM (PCRAM)	small cell size, high scalability	write endurance, write disturbance, resistance drift
Spin-Torque Transfer RAM (STTRAM)	small cell size, efficient read operations, high scalability	asymmetric read and write energy
Resistive RAM (RRAM)	low-energy writes, high data density	write endurance

TABLE I: Advantages and drawbacks of PCRAM, STTRAM, and RRAM.

A. Phase Change RAM

PCRAM works by employing a phase change material that is sensitive to heat to store data. Heat is generated through electrical pulses that are sent to a PCRAM cell to either melt it (RESET) or crystallize it (SET). PCRAM has a smaller cell size and higher scalability over DRAM with PCRAM chips being constructed at a 20 nm technology node. PCRAM also has no leakage power from the cell [25]. All of these factors–though primarily energy efficiency–have led to PCRAM being projected to replace DRAM in main memory.

Though PCRAM appears to be the most promising technology for volume production, there are critical issues standing in its way. One key issue is scaling. At 54 nm PCRAM, a phenomenon referred to as write-disturbance (WD) is observed, which refers to the disturbance of nearby cells resistance-states from the heat generated for writing one cell. This phenomenon becomes more significant at smaller dimensions, especially below 20 nm technology [25].

Another roadblock is write endurance. "Stuck at faults" can occur after a limited number of writes (10^7–10^8). When this occurs, the state is still discernible; however, it cannot be overwritten. Limited solutions to this problem do exist, such as excluding memory pages with faulty bits or relying on the operating system to hide the faults; however, these solutions are not ideal. For example, the former continuously reduces usable memory until, presumably, no working cells remain [26].

B. Spin-Torque Transfer RAM

In contrast to PCRAM, STTRAM uses magnetism to store data. An STTRAM storage element, referred to as a magnetic tunnel junction (MTJ), contains two ferromagnetic layers with a tunnel barrier between. One layer is used as a reference layer that is fixed at fabrication and data is represented by the difference in magnetization direction between the two layers. Two possible states exist: parallel directions and nonparallel directions.

STTRAM also has its hurdles. Perhaps the biggest is STTRAM cache write performance. When compared to current SRAM-based caches, STTRAM caches consume 2–3$\times$ more energy for writes and have a longer write latency. Also, read and write energy is highly asymmetric with write energy about an order of magnitude higher than read energy [9]. Improving inefficient STTRAM cache write operations is essential for adoption of STTRAM in LLCs.

C. Resistive RAM

RRAM refers to the subset of resistive NVMs that use metal oxides to store data. A metal-oxide RRAM cell consists of two layers of metal electrodes with a metal oxide layer in the middle. In order to change the resistance or state of the cell, an external voltage with a specified polarity, magnitude, and duration is applied to the metal-oxide layer [27].

RRAM has superior write endurance to PCRAM with issues occurring at 10^{10} writes rather than 10^7 to 10^8 writes. Additionally, RRAM can still benefit from write endurance solutions for PCRAM [27]. RRAMs can also be implemented with a unique dense crossbar architecture which means that these RRAMs cannot benefit from architecture-level optimizations for other NVMs since other NVMs often use a MOSFET or BJT as an access transistor. However, the RRAM

crossbar architecture has potential for unique, crossbar-specific innovations [27].

Micron has produced a RRAM prototype and HP has its Memristor crossbar project, both indicating industry interest in RRAM for main memory [27].

D. Qualitative Comparison

PCRAM boasts a small cell size and high scalability, however it suffers from poor write endurance, write disturbance, and resistance drift. STTRAM allows for highly efficient read operations and high scalability; but this comes with highly asymmetric read and write energy. RRAM stands apart from STTRAM in that it provides low-energy writes. It also can provide very high data density, however, this comes with the cost of poor write endurance. These three byte-addressable NVMs can have multiple levels per cell for increased data density and memory capacity [37].

Cell-level modeling parameters for each NVM are shown in Table II. Each NVM has its own unique characteristics, specifically the degree of asymmetry. Consequently, these characteristics have certain implications for area, performance (execution time), and energy consumption.

E. Related Work

Related work [38], [39] which models emerging NVM technologies such as PCRAM, STTRAM, and RRAM in the LLC often does not include the device-level parameters which were used in the models. In other cases, only a subset of the parameters are released. In fact, in extreme cases [6], works use a rough model of an NVM-based LLC simply by adding a delay to the write latency of SRAM and scaling the SRAM capacity. Determining model parameters used in an NVM simulator is a critical step in generating an accurate model and is non-trivial. Without knowledge of the methodology that is used, it is impossible to obtain apples-to-apples comparisons across works. This is a significant hindrance to the overall goals of these works.

In some related work, these parameters are based on observations from a real chip, in which case it is obviously appropriate to leave out the methodology for determining these parameters. In many cases, though, these models are based on NVM technologies introduced in VLSI literature which often do not contain the architecture parameters to complete a specification in an NVM simulator. Thus it is important for researchers to release this information. While NVM simulators are continuously being improved, they are still in their early stages. Therefore, in concert with simulator development, it is prudent to also develop better methodologies for most effectively utilizing these simulators. This work gains part of its novelty by modeling a wide range of NVM technologies–across class and generations within class–in a transparent manner.

III. Cell-level NVM Models

As mentioned previously, NVM-based LLC models in prior art are generally course-grained and opaque. Furthermore, few works compare across multiple NVM technologies. The lack of a consistent modeling methodology or set of models prevents researchers from easily making apples-to-apples comparisons across technologies. This is clear yet no such methodology nor set of models exists for NVM-based LLCs. This is, in fact, because it is challenging to derive an accurate NVM LLC model. This is primarily due to:

- Limited number of architectural NVM simulators available
- Irregularities in the kind of data presented in VLSI literature works which introduce NVM technologies
- Lack of direct compatibility between presented data and the required parameters for the simulators

To date, there are a limited number of software tools available for modeling NVMs. Among them are NVSim [40], NVMain [41], and DESTINY [42]. Each of these NVM simulators are slightly tangential. One may model energy and timing, while another models area, energy, and lifetime, for example. Multiple NVM simulators can be used together to generate all the desirable estimates (timing, energy, area, and lifetime) but this is very tedious. For this work, we rely on NVSim, since it is currently the most established NVM simulator. We use NVSim to generate timing, energy, and area for NVM-based LLCs containing the NVM cell specifications shown in Table II.

In addition to the limited number of (tangential) NVM simulators, it is difficult to obtain the data necessary to effectively use the simulators. To model an NVM-based LLC in NVSim, certain parameters are required depending on the class. For example for PCRAM, NVSim requests the process node and cell size followed by parameters for reading and writing the device: read current, read energy, reset current, reset pulse, set current, and set pulse. For STTRAM, NVSim requires process node, cell size, read voltage, read power, reset current, reset pulse, reset energy, set current, set pulse, and set energy. RRAM requires process node, cell size, read voltage, read power, reset voltage, reset pulse, reset energy, set voltage, set pulse, and set energy.

For the NVMs we have selected, these parameters and values are listed in Table II. For ease of communication, NVM cells will be referred to by the citation name for the rest of this section.

A. Modeling Heuristics

It is difficult to judge the absolute accuracy of any NVM model derived from the VLSI literature; however, this is a common modeling strategy for NVM-based LLC works. This leads to apples-to-oranges comparisons across NVM-based LLCs used in different works. Since some necessary NVM cell-level parameters may not be available, specifying a technology in a simulator such as NVSim requires educated guesses for those parameters which are not found in the literature. These parameters which were not reported in the literature for any of the NVMs in this work are indicated with an * or † in Table II. The most common data points which are not reported in the original VLSI paper, yet are necessary

	Oh [28]	Chen [29]	Kang [30]	Close [31]	Chung [32]	Jan [33]	Umeki [34]	Xue [35]	Hayakawa [36]	Zhang [13]
class	PCRAM	PCRAM	PCRAM	PCRAM	STTRAM	STTRAM	STTRAM	STTRAM	RRAM	RRAM
year	2005	2006	2006	2013	2010	2014	2015	2016	2015	2016
access device	CMOS	CMOS	CMOS	CMOS	CMOS	CMOS	CMOS	CMOS	CMOS	CMOS
process $[nm]$	120	60*	100	90	54	90	65	45	40	22
cell size $[F^2]$	16.6*	10*	16.6	25	14	50	$48^{\dagger}$	63	4*	4*
cell levels	1	1	1	2	1	1	1	2	1	1
read current $[\mu A]$	40*	40*	60*	60*						
read voltage $[V]$					0.65	0.08	0.38	1.2	0.4*	0.2
read power $[\mu W]$					$24.1^{\dagger}$	30*	1.70	65	0.16*	0.02
read energy $[pJ]$	2*	2*	2*	2*						
reset current $[\mu A]$	600	90	600	400	80	52	$255^{\dagger}$	150		
reset voltage $[V]$									2*	1
reset pulse $[ns]$	10	60	50	20	10	4	10	2	10*	150
reset energy $[pJ]$					$0.52^{\dagger}$	1*	1.12	0.36	0.6*	0.4
set current $[\mu A]$	200	55	200*	400	$100^{\dagger}$	38	$255^{\dagger}$	150		
set voltage $[V]$									2*	1
set pulse $[ns]$	180	80	300	20	10	4.5	10	2	10*	150
set energy $[pJ]$					$0.75^{\dagger}$	1*	1.12	0.36	0.6*	0.4

TABLE II: NVM Parameters († and * indicate parameters not found in cited paper. † denotes parameters derived using heuristic 1). * denotes parameters derived using heuristic 2) or 3). Grayed-out cells indicate parameters not applicable to that class).

for architectural simulation include: read current, read energy, read power, reset energy, and set energy.

Without access to industry data, it is a challenging exercise to validate the absolute accuracy of all cell-level parameters; however, it is not essential for this work. To ensure an apples-to-apples comparison across different technologies, a consistent set of heuristics must be used to select values for parameters which are not available. We developed such heuristics, and it includes three strategies:

1) *Electrical Properties*: Derive unknown parameters based on known parameters (e.g., equations (1)–(3), where $_{s/r}$ denotes set or reset, t denotes pulse length, A denotes cell size, and s denotes process size, e.g., 45 nm). These parameters are denoted with a † in Table II.
2) *Interpolation*: Use trends of known parameters from similar technologies to interpolate unknown parameters from known parameters.
3) *Similarity*: Select parameter from another technology in the same class.

Values in Table II which contain a † are unknown parameters whose estimates were obtained using heuristic 1). This is the most accurate method of estimation and should be preferred. For heuristics 2) and 3), parameters from an older technology in the same class are used to estimate unknown parameters. Parameters obtained this way are denoted with a *.

To demonstrate the application of these heuristics, we present an example using heuristic 3), the least accurate (and least preferred) method for determining the value of an unknown parameter. Using the knowledge of the set current and reset current of *Oh*, a technology in the same class as *Kang*, we select the set current for *Kang*. Given that *Oh* and *Kang* have identical reset current, it is likely that they have identical, or at least similar, set current. Therefore we select a set current of 200 μA for *Kang*.

Due to the more recent nature of RRAM, there are fewer CMOS-accessed RRAM technologies with cell-level parameters for specification in NVSim. Despite the lack of parameters in the literature for Hayakawa [36], we include it in this work to maintain a more balanced set of NVM classes. We should note: upon publication all of our NVM cell models will be made publicly available online. They can be downloaded here: http://sites.tufts.edu/tcal/nvm-models. Using these models and equations (4)–(8) in Section IV, it is straightforward to derive NVM-based LLC models based on these NVM cells which can directly interface with a full-system simulator (see Section IV).

$$P_{read} = I_{read} * V_{read} \tag{1}$$

$$E_{s/r} = I_{s/r} * V_{access} * t_{s/r} \tag{2}$$

$$A\,[F^2] = \frac{l_{cell} * w_{cell}}{s_{proc}^2} \tag{3}$$

IV. Evaluation Methodology

For our evaluation, we model a quad-core out-of-order architecture based on the Xeon x5550 "Gainestown" processor using an open source x86 simulator, Sniper [43]. Each core is clocked at 2.66 GHz with 1 thread per core. We use a 128-entry reorder buffer with a 48 entry load queue and 32 entry store queue. The architecture is equipped with three levels of cache. Each core contains its own 32KB, 4-way set associative, parallel access, write-back L1 instruction cache and similar 8-way set associative L1 data cache. Each core also contains a private 256KB, 8-way set associative, parallel access, write-back L2 cache. All cores share an on-chip LLC capacity of 2MB. The LLC model in Sniper is modified to handle both SRAM and NVM (PCRAM, STTRAM, and RRAM). Our baseline configuration contains a 2MB SRAM-based LLC with a 45 nm process technology. The latency, energy, area, and capacity of all NVM-based LLCs is shown in Table III. We sweep LLC memory technology and study the impact of NVM behavior on system performance and energy.

Our main memory model consists of 4 distributed DRAM controllers with 4 DIMMS per controller and 104K entries

	$\mathbf{Oh_P}$	$\mathbf{Chen_P}$	$\mathbf{Kang_P}$	$\mathbf{Close_P}$	$\mathbf{Chung_S}$	$\mathbf{Jan_S}$	$\mathbf{Umeki_S}$	$\mathbf{Xue_S}$	$\mathbf{Hayakawa_R}$	$\mathbf{Zhang_R}$	SRAM
Area $[mm^2]$	6.847	4.104	4.591	2.855	1.452	9.171	4.348	1.585	0.915	0.307	6.548
Tag Access Latency $[ns]$	0.74	0.604	0.656	0.582	1.240	1.423	1.208	1.156	1.396	1.722	0.439
Data Read Latency, t_{read} $[ns]$	1.907	0.607	1.497	0.82	1.763	3.072	2.715	2.878	1.722	2.16	1.234
Data Write Latency, t_{write} $[ns]$ (set/ reset)	181.206 /11.206	80.491 /60.491	301.018 /51.018	20.681 /20.681	11.751	7.878	11.916	4.038	20.716	300.834	0.515
Cache Hit Dynamic Energy, $E_{dyn,hit}$ $[nJ]$	0.840	0.421	0.678	0.437	0.209	0.188	0.173	0.251	0.263	0.217	0.565
Cache Miss Dynamic Energy, $E_{dyn,miss}$ $[nJ]$	0.042	0.025	0.033	0.023	0.082	0.077	0.058	0.121	0.078	0.086	0.011
Cache Write Dynamic Energy $E_{dyn,write}$ $[nJ]$	225.413	34.108	375.073	51.116	1.332	2.305	1.644	0.597	0.952	0.523	0.537
Cache Total Leakage Power $[W]$	0.062	0.071	0.061	0.039	0.166	0.048	0.295	0.115	0.194	0.151	3.438

Capacity [MB]	2	4	2	4	8	1	2	8	32	128	2
Tag Access Latency $[ns]$	0.740	0.607	0.656	0.581	1.283	1.288	1.208	1.229	1.690	2.392	0.439
Data Read Latency, t_{read} $[ns]$	1.909	1.428	1.497	0.789	3.262	2.074	2.715	3.378	2.536	9.537	1.234
Data Write Latency, t_{write} $[ns]$ (set/ reset)	181.206 11.206	81.17/ 61.17	301.018/ 51.018	20.46/ 20.46	13.088	6.17	11.916	3.928	20.735	304.936	0.515
Cache Hit Dynamic Energy, $E_{dyn,hit}$ $[nJ]$	0.840	0.496	0.678	1.003	0.457	0.187	0.173	0.683	0.715	0.605	0.565
Cache Miss Dynamic Energy, $E_{dyn,miss}$ $[nJ]$	0.042	0.030	0.033	0.029	0.083	0.080	0.058	0.123	0.088	0.089	0.011
Cache Write Dynamic Energy, $E_{dyn,write}$ $[nJ]$	225.413	33.599	375.073	50.912	1.656	1.780	1.644	0.912	1.458	0.921	0.537
Cache Total Leakage Power $[W]$	0.062	0.100	0.061	0.137	0.661	0.025	0.295	0.828	3.896	9.000	3.438

TABLE III: Gainestown LLC models generated by NVSim for *fixed-capacity* (top) LLC, *fixed-area* (bottom) LLC. For PCRAM, data write latency format is: set/ reset. Heatmap on per-row basis indicating extrema.

per directory. Each directory is full-map and each controller can provide bandwidth up to 7.6 GB/s. The full simulated architecture details are shown in Table IV.

We select workloads from four benchmark suites: SPEC cpu2006 [44], PARSEC 3.0 [45], NAS Parallel Benchmarks (NPB) 3.3.1 [46], and SPEC cpu2017 [44]. Benchmark suites were selected to provide a range of application domains and parallelism. The PARSEC3.0 workloads include image processing and video encoding kernels, the cpu2006 and NPB3.3.1 workloads contain computer science, scientific, and mathematical kernels, and the remaining workloads are the AI inference benchmarks from cpu2017. The PARSEC3.0 and NPB 3.3.1 suites contain multi-threaded applications while cpu2006 and cpu2017 contain single-threaded workloads. We select 7 from cpu2006, 2 from PARSEC3.0, 8 from NPB 3.3.1, and 3 from cpu2017. Workloads were selected from each benchmark suite which exhibit an LLC misses per thousand-instructions (mpki) at least greater than 5 to appropriately stress the LLC. Statistical inference use cases, like the ones included from cpu2017, were selected to expose the NVM-based LLC system to application behavior which is likely to be prominent in emerging and future workloads. We should note: the cpu2017 AI workloads (*deepsjeng*, *leela*, *exchange2*) are examples and may not be representative of all AI workloads. *deepsjeng* performs alpha-beta tree search, *leela* contains Monte Carlo tree search, and *exchange2* is a recursive algorithm. There are other AI benchmark suites which are now available like Fathom [47] and TBD [48], which are more focused on deep learning tasks. Our complete list of benchmarks is shown in Table V.

Simulated Architecture	
μprocessor	Xeon x5550 "Gainestown" 2.66 GHz OoO Quad-core, 1 thread/core
ROB	128-entry ROB, 48-entry load queue, 32-entry store queue, store-to-load forwarding, and ROB repartitioning
L1I $	private, 32KB, 4-way set associative, parallel-access, write-back
L1D $	private, 32KB, 8-way set associative, parallel-access, write-back
L2 $	private, 256KB, 8-way set associative, parallel-access, write-back
L3 $	shared, 2MB, 64B blocks, 16-way set associative, parallel-access, write-back
DRAM	104K entries/directory controller, associativity=16, full-map directory, 4 distributed DRAM controllers, 7.6GB/s per controller, 8 chips/dimm, 4 dimms/controller

TABLE IV: Simulated Architectural Parameters

A. NVM-based LLC Model in Sniper

Due to the asymmetric nature of NVM energy and latency for read and write operations, it is non-trivial to model such behavior in a standard x86 simulator, which is agnostic to read or write. To model an asymmetric, parallel-access NVM-based LLC in Sniper, we use equations (4)–(8).

$$t_{read} \sim 2 * t_{H-tree} + t_{read,mat} \tag{4}$$

$$t_{write} \sim 1 * t_{H-tree} + t_{write,mat} \tag{5}$$

Equations (4) and (5) show the approximation of data read latency, t_{read}, and data write latency, t_{write} for our LLCs with H-tree routing, where t_{H-tree} is the H-tree routing latency and $t_{read/write,mat}$ is the read/write latency within a mat. t_{H-tree} and $t_{read/write,mat}$ are direct outputs from NVSim.

$$E_{dyn,\ hit} = E_{dyn,\ tag} + E_{dyn,\ data-read} \tag{6}$$

Suite	Bmk	LLC mpki	Description
cpu2006	bzip2	142.69	Compression/Decompression, s.t.
cpu2006	gamess	12.83	Quantum computations, s.t.
cpu2006	GemsFDTD	12.56	Maxwell solver 3D, s.t.
cpu2006	gobmk	38.08	Plays Go and analyzes, s.t.
cpu2006	milc	16.46	Lattice gauge theory, s.t., MIMD
cpu2006	perlbench	7.57	Perl interpreter s.t.
cpu2006	tonto	12.39	Quantum package, s.t.
PARSEC3.0	x264	17.81	MPEG-4 encoding, s.t.
PARSEC3.0	vips	5.43	Image transformation, m.t.
NPB 3.3.1	cg	80.89	Conjugate gradient, m.t.
NPB 3.3.1	ep	9.31	Embarrassingly parallel, m.t.
NPB 3.3.1	ft	15.39	discrete 3D FFT, m.t.
NPB 3.3.1	is	35.63	Integer sort, m.t.
NPB 3.3.1	lu	14.42	LU Gauss-Seidel solver, m.t.
NPB 3.3.1	mg	65.09	Multigrid on meshes, m.t.
NPB 3.3.1	sp	44.35	Scalar penta-diagonal solver, m.t.
NPB 3.3.1	ua	39.08	Unstructured adaptive mesh, m.t.
cpu2017	deepsjeng	159.58	AI: alpha-beta tree search, s.t.
cpu2017	leela	24.05	AI: Monte Carlo tree search, s.t.
cpu2017	exchange2	13.50	AI: recursive solution generator, s.t.

TABLE V: Summary of workloads used [49]. 's.t.' refers to *single-threaded* and 'm.t.' refers to *multi-threaded*

$$E_{dyn,\ miss} = E_{dyn,\ tag} \tag{7}$$

$$E_{dyn,\ write} = E_{dyn,\ tag} + E_{dyn,\ data-write} \tag{8}$$

Equations (6)–(8) show the calculation of dynamic energy for the LLC, where $E_{dyn,\ hit}$ is the dynamic energy for a hit, $E_{dyn,\ tag}$ is the dynamic energy for a tag lookup, $E_{dyn,\ data-read}$ is the data read dynamic energy, $E_{dyn,\ miss}$ is the dynamic energy for a miss, $E_{dyn,\ write}$ is the dynamic energy of an LLC write, and $E_{dyn,\ data-write}$ is the data write dynamic energy. This data is shown in Table III for all LLCs for *fixed-capacity* and *fixed-area* (these two configurations are explained in Section IV-C). In the table, and for the remainder of the work, NVM-based LLCs will be referred to by their citation name *plus a subscript indicating the class*. For example, *Zhang*$_{\mathrm{R}}$ indicates a RRAM technology.

B. Workload Characterization

To understand why particular NVMs perform better or worse depending on the application, we used PRISM [50] to conduct a rigorous analysis of the memory characteristics of our applications. PRISM is a framework for profiling workloads to obtain architecture-agnostic workload metrics. This framework allows us to capture virtual memory addresses accessed in the execution of a workload. The first metric we use, *memory entropy*, represents the randomness of accessed memory addresses. We capture two forms of memory entropy: global memory entropy and local memory entropy. Global memory entropy represents the temporal locality of all memory accesses. Local memory entropy represents the spatial locality of regions of the memory address space. This is computed by skipping the *M lowest order bits* of the address in the entropy computation; we chose M to be 10 to reflect page size. In Information Theory, entropy is computed with the Shannon entropy equation [24] (equation 9). Applying the Shannon entropy equation to memory addresses: $p(x_i)$ is the probability of x_i, representative of how frequent a memory address appears in an application, while N is the total number of addresses, and x_i is each address. The resulting H is the memory entropy, in terms of bits. For this work, we compute memory entropy for reads and writes separately. By splitting up the memory accesses into reads and writes, we aim to characterize the effect of the asymmetric operation of NVMs.

$$H = -\sum_{i=1}^{N} p(x_i) * log_2(p(x_i)) \tag{9}$$

The second metric, unique reads and writes, is representative of the memory address space during the execution of an application. It is generated by logging a running, unique set of addresses as we iterate over each per-thread trace. This metric is again computed separately for read accesses and write accesses.

The third metric, 90% memory footprint, is an estimate of the working set of the application. It is calculated by logging the number of accesses per memory address. The addresses are then ordered, descending, by the number of times they are accessed. We count the number of unique addresses, starting from the most accessed address, up to 90% of *all* memory accesses.

For our workload characterization analysis, we exclude four benchmarks from cpu2006: *gamess, gobmk, milc, and perlbench* due to incompatibilities with PRISM.

C. Fixed-Capacity vs Fixed-Area

Given that density is a primary benefit of NVMs, we consider this effect in our simulations by employing two dfferent simulation strategies. We refer to these two strategies as *fixed-capacity* and *fixed-area*. In the fixed capacity configuration, we model our NVM-based LLCs to have the same capacity as the baseline SRAM-based LLC model. This assumes that the architecture is *cost-limited*, i.e., a reduction in chip-area is favorable and LLC capacity is not a bottleneck. In the second configuration, fixed-area, we assume the architecture is capacity-limited, i.e., LLC capacity is the bottleneck, and more capacity is desired with the assumption that physical area will not exceed that of SRAM. We believe industry decisions primarily fall into the *fixed-area* category, especially with the advent of ML and AI use cases; however, we analyze performance and energy for both setups.

V. Simulation Results and Analysis

In this section, we analyze the performance and energy of our simulated architecture with the NVM-based LLCs and the baseline configuration. We organize our results and analysis into the two categories, based on our two aforementioned simulation configurations: *fixed-capacity* and *fixed-area*. Again, *fixed-capacity* assumes that we have a particular capacity budget, while *fixed-area* assumes we have an area budget but are not limited on capacity. For each NVM LLC, we study performance and energy, and compare to the SRAM baseline. To evaluate performance and energy of the NVM-based LLCs, we report overall system speedup, LLC total

energy, and ED^2P. Each of these is normalized to the baseline SRAM configuration.

The rest of this section is laid out as follows: Section V-A describes the performance and energy of the NVM-based LLCs compared to the SRAM baseline for both the single-threaded and multi-threaded use cases in the *fixed-capacity* configuration. We then characterize the variation in performance and power across different use cases. Section V-B describes the performance and energy of the NVM-based LLCs compared to the SRAM baseline for both the single-threaded and multi-threaded use cases in the *fixed-area* configuration. Again, we characterize the observed variation across memory technology. In Section V-C, we present a sensitivity study of multi-core systems by performing a detailed analysis of performance and power of multi-core systems with NVM-based LLC compared to a baseline single-core SRAM-based LLC architecture. We characterize the unique tradeoffs involved with NVM-based LLCs as number of cores/threads increases. For ease of communication in the rest of this section, we will refer to NVM-based LLCs by their respective citation name plus class subscript (from Table III).

A. *Fixed-Capacity*

Figures 1a and 1b show speedup, LLC energy, and ED^2P for the fixed-capacity configuration. Figure 1a shows those for the single-threaded workloads and Figure 1b shows those results for the multi-threaded workloads.

1) Single-threaded Performance: Utilizing an NVM LLC generally results in a loss in performance neighboring -1% to -3%. The top plot in Figure 1a reports normalized speedup. We should note that in some cases, NVM performance is identical to that of SRAM, and in one case, superior. For *deepsjeng*, NVM speedup is on average +3%.

2) Single-threaded Energy: The middle plot in Figure 1a reports LLC energy normalized to the SRAM baseline. Shorter bars represents better energy savings. The horizontal line indicates LLC energy for the SRAM baseline. NVM LLC energy is up to $10\times$ less than SRAM in most cases. $Kang_P$ and Oh_P, two PCRAM technologies, exhibit worst case LLC energy–up to $6\times$ more than SRAM.

3) Single-threaded ED^2P: Higher speedup means higher energy consumption for the LLCs. The last plot in Figure 1a reports normalized ED^2P. A smaller ED^2P is better. NVM LLC ED^2P is up to $10\times$ less than SRAM in most cases. Similar to the LLC energy results, two PCRAM technologies, $Kang_P$ and Oh_P, exhibit worst-case ED^2P. This is for *bzip2*, *deepsjeng*, *gobmk*, and *leela*; this includes two out of the three AI workloads.

4) Multi-threaded Performance: Multi-threaded performance is mostly agnostic to LLC technology. The top plot in Figure 1b reports normalized speedup. We should note that there is a slight (2%) average speedup for *cg* with NVM LLC. For *is* and *lu*, however, performance degrades by up to 10%.

5) Multi-threaded Energy: The middle plot in Figure 1b reports normalized LLC energy. NVM LLC energy is $8-10\times$ less than SRAM for most workloads. Worst-case NVM energy

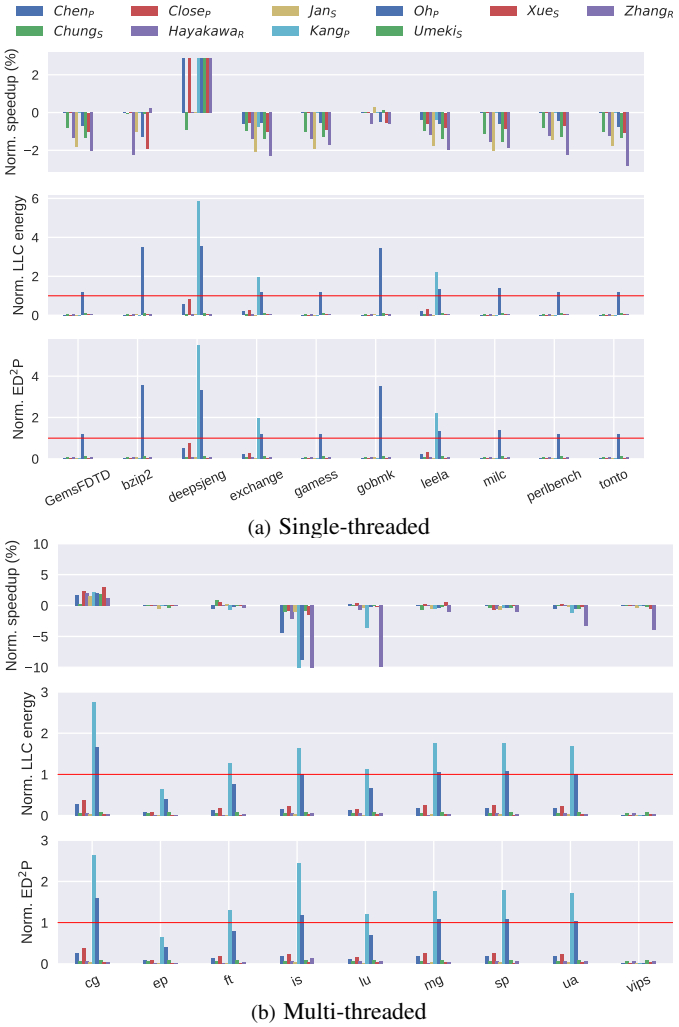
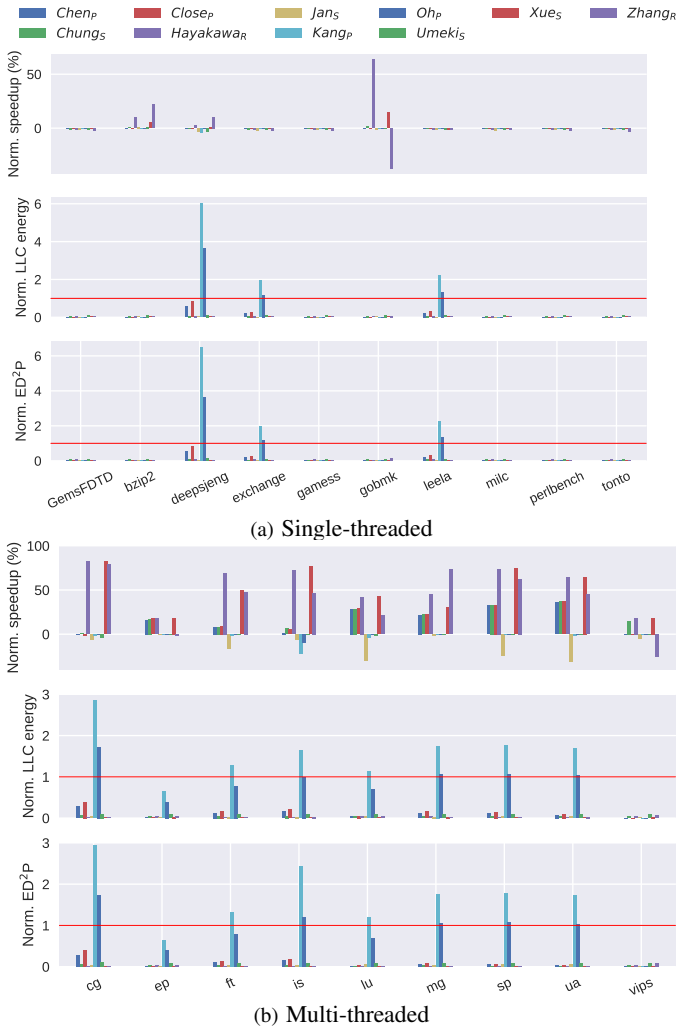


Fig. 1: Simulation results for Gainestown with *fixed-capacity* LLC

is reported to be $2.8\times$ SRAM. This is for *cg* with the PCRAM technology, $Kang_P$. $Kang_P$ also exhibits poor energy efficiency with *is*, *mg*, *sp*, and *ua*.

6) Multi-threaded ED^2P: The last plot in Figure 1b reports normalized ED^2P. We note that NVM ED^2P is superior to SRAM for virtually all cases of NVM technology and workload. $Kang_P$ and Oh_P ED^2P is similar to SRAM in the worst case.

7) **Summary:** *Fixed-capacity* performance for both single-threaded and multi-threaded workloads is consistent with SRAM. Despite worse write latency of NVM cells, this excess latency may not show up in system execution time. This is due to an assumption in the simulator that LLC writes happen off of the critical path. Without this, exceptionally high write latency could more significantly impact system execution time. The most energy-efficient NVM is Jan_S. Jan_S outperforms all memory technologies for *gamess*, *GemsFDTD*, *milc*, *perl*, *tonto*, *leela*, *exchange2*, *vips*, *x264*, and all NPB workloads, except for *ep*. For *vips*, *x264*, and *sp*, this improvement in



(a) Single-threaded

(b) Multi-threaded

Fig. 2: Simulation results for Gainestown with *fixed-area* LLC

energy efficiency is 50% over the next best. Jan_S performs anywhere between 20% and 50% over the next most energy efficient technology depending on the use case. For *bzip2*, *gobmk*, and *deepsjeng*, Xue_S outperforms all memory technologies in energy efficiency by 10% over the next best. The results are similar for ED^2P.

B. Fixed-Area

Cache area of each LLC with a fixed capacity to that of the baseline SRAM LLC is shown in Table III. MLC NVMs result in significant area savings. Xue_S (2 levels per cell) has an area of $3.8mm^2$ while Jan_S has area of almost an order of magnitude greater. RRAM technologies outperform both STTRAM and PCRAM in area, with $Zhang_R$ area of just $0.3mm^2$ for a 2MB cache.

NVM cache capacities for a given cache area budget are shown in Table III. For this table, cache area was fixed at 6.55 mm^2. The results from Table III are reinforced with $Zhang_R$ having a massive cache capacity for the given area budget. STTRAM LLCs are also dense. Xue_S reaches about 8MB, $4\times$ that of its SRAM counterpart. These fixed-area NVM caches were simulated in the same way as their fixed-capacity counterparts and the results are shown in Figure 2b.

Speedup, LLC energy, and ED^2P for the fixed-area configuration are shown in Figures 2a and 2b. Figure 2a shows those for the single-threaded workloads and Figure 2b shows those results for the multi-threaded workloads.

1) Single-threaded Performance: The top plot in Figure 2a reports speedup for the single-threaded workloads. On average, speedup is negligible. For *gobmk*, $Hayakawa_R$ achieves a 60% speedup over the SRAM baseline. In the case of *bzip2*, $Zhang_R$ results in a 20% increase in overall system speedup, while in the case of *gobmk*, reduces performance by 40%.

2) Single-threaded Energy: For fixed-area, LLC energy savings are generally upwards of $10\times$. The middle plot in Figure 2a shows the LLC energy results. We should note that: for $Kang_P$ and Oh_P, these NVMs exhibit poor energy efficiency for the AI use cases.

3) Single-threaded ED^2P: The bottom plot in Figure 2a reports ED^2P. ED^2P results show similar characteristics to LLC energy results. The two PCRAM technologies, $Kang_P$ and Oh_P exhibit the worst ED^2P, and this is shown for the AI workloads, *leela*, *deepsjeng*, and *exchange2*.

4) Multi-threaded Performance: Performance varies widely depending on workload and NVM technology. The first plot in Figure 2b reports normalized speedup. For the *ep*, *lu*, *mg*, *sp*, and *ua* workloads, $Chen_P$, $Chung_S$, $Close_P$, $Hayakawa_R$, and $Zhang_R$ exhibit more than a 10% speedup over the baseline. For the *ft*, *lu*, *sp*, and *ua* workloads, Jan_S exhibits more than a 10% *reduction* in performance.

5) Multi-threaded Energy: The second plot in Figure 2b shows normalized LLC energy. LLC energy is generally stable. Again, the two PCRAM technologies, $Kang_P$ and Oh_P, exhibit especially poor energy efficiency (up to $3\times$).

6) Multi-threaded ED^2P: The last plot in Figure 2b reports normalized ED^2P. ED^2P results show similar characteristics with LLC energy results.

7) **Summary:** Poor write latency is often off the critical path and hidden from overall system performance. For the *gobmk* and *ft* workloads, $Hayakawa_R$ achieves 40% and 60% increase in speedup, respectively, over the next best technology. The most energy efficient technology is highly dependent on use case. For *gobmk*, $Hayakawa_R$ outperforms all technologies in energy efficiency with 20% increase in efficiency over the next best. For the *gamess*, *GemsFDTD*, *milc*, *perl*, *tonto*, *leela*, *exchange2*, *vips*, *x264*, and *ep* workloads, Jan_S outperforms all technologies by a 30% improvement in energy efficiency over the next best technologies. For the *cg*, *is*, *lu*, *sp*, and *ua* workloads from NPB, Xue_S exhibits superior energy efficiency ranging from 30% to 50% over the next best.

ED^2P exhibits similar characteristics to LLC energy. We should note a few key differences. For the *gobmk* workload, $Hayakawa_R$ is 90% better than the next best NVM. Further, for the *ft* workload, $Hayakawa_R$ exhibits a 60% improvement in ED^2P over the next best. For the *vips* and *x264* bench-

marks, Xue_S achieves a 20% improvement over the next best technologies.

C. Sensitive Study: Core Sweep

In this section we present a sensitivity study with multi-core systems by performing a detailed analysis of performance and power of multi-core systems with NVM-based LLC compared to a baseline single-core SRAM-based LLC architecture. We characterize the unique tradeoffs involved with NVM-based LLCs as number of cores/threads increases.

1) Performance: Capacity is an increasing strain on the systems as cores increase. For *fixed area*, the best performing NVMs are those with high capacity and low latency. Large capacity mitigates thread starvation from sharing an LLC among more cores. If starvation occurs, low latency reduces the write penalty. Xue_S and $Hayakawa_R$ exhibit this behavior. The lower latency of a dense Xue_S makes up for an increase in misses. For capacity starved benchmarks, such as *mg*, $Zhang_R$ and $Hayakawa_R$ show the best performance as they are the densest (128MB and 32MB).

For *ft* Xue_S has a low write latency, but suffers from lower density. $Hayakawa_R$ has a higher write latency, but a 4× increase in capacity mitigates this. $Zhang_R$ has an even larger capacity, but a nearly 15× worse write latency than Hayakawa_R, stripping away the benefit of its higher capacity. Such behavior is represented with the *cg*, *lu*, and *sp* workloads.

2) Power: Leakage matters more with slower runtimes, which depends on both capacity and latency. $Umeki_S$ is one of the slowest NVMs due to capacity throttling its performance. $Umeki_S$ has the worst energy efficiency for an NVM. Despite its lower leakage than other technologies like $Zhang_R$, Xue_S, and $Hayakawa_R$, the system with $Umeki_S$ LLC takes longer to complete and thus has more time to leak. Comparatively, $Hayakawa_R$ has 16× the capacity while also having a 2× longer write latency yet it is one of the highest performing. We should note that for 4-32 cores running *ft* and all core configurations running *sp*, $Hayakawa_R$ is the most energy efficient despite having 13× greater leakage power than $Umeki_S$.

Jan_S is a top NVM for energy due to its drastically lower leakage power than the others. It is 32× less than Xue_S, 156× less than $Hayakawa_R$, and 360× less than $Zhang_R$. However, its execution time is much worse than all other memories. For workloads that do not take excessively long for systems with Jan_S LLC to complete, it is possible for it to have better energy usage. For runs such as: 4+ cores running *ft*, all multicore configurations running *cg*, 8+ cores running *lu* and all configurations running *sp*, the slow run time of Jan_S canceled out its dynamic energy efficiency. Leakage is why Xue_S outperforms $Zhang_R$ for energy. Although they have similar dynamic statistics, $Zhang_R$ suffers from a 11.25× higher leakage power than Xue_S. In addition, it usually takes longer than Xue_S to complete; running *mg* is an exception to this.

The extent to which dynamic energy plays a role in increasing total energy consumption is at odds with the savings from reduced static power usage. This is either from the NVM strictly leaking less or from the reduced total leakage because of a shorter runtime. Between the fastest NVMs like Xue_S, $Hayakawa_R$, and $Zhang_R$, only Xue_S and $Zhang_R$ are most energy efficient. This is due to the higher dynamic energy of $Hayakawa_R$. The benefit of Xue_S over $Zhang_R$ in this case is due to the higher leakage of $Zhang_R$.

VI. Correlation with Workload Features

In this section, we present our workload characterization framework. We use the performance and energy results from the previous section to determine how architecture-agnostic behavior affects performance and energy. To begin, we present our workload characterization data and give a brief overview. Next, we present our framework. We then present results for two kinds of systems: a *general purpose* system and a *specialized system*. For the general purpose case, we consider all use cases together. For the specialized case, we present results for a specific application domain–AI.

For our workload characterization, we use a rigorous set of memory behavior metrics broken down by reads and writes. These metrics are shown in Table VI. For each use case, we compute global entropy, local entropy, unique address footprint, 90% address footprint, and total address footprint (for a definition of these metrics, see Section IV). Table VI lists the data for each benchmark. Each column in Table VI is an individual heat map to more easily see variation in behavior across use case. For *GemsFDTD*, we notice a 90% read and write address footprint that is two orders of magnitude greater than all other use cases. For *exchange2*, *x264*, and *lu*, we observe a total read address footprint that is an order of magnitude greater than all other workloads. *x264* and *lu* are significantly read heavy. While *exchange2* has the largest total read and write address footprint, it has the smallest unique read and write address footprint, two orders of magnitude smaller than the average use case.

From the simulation results, it is seen that performance and energy vary across both use case and NVM without a clear correlation between performance and energy and the characteristics of the NVM technology. This complicates the task of selecting the optimal NVM technology for a given use case. Since different NVMs, even within the same class of technology, exhibit unique latency and energy characteristics, certain technologies lend themselves better to different use case behaviors. From the workload characterization data, we observed that metrics like working set size, spatial locality, and data footprint can vary vastly across application domain. In fact, use case memory access behavior varies among individual use cases within a single domain.

To better understand the relationship between NVM and application behavior, we develop a mathematical framework for doing so. We use linear correlation between architecture-agnostic use case features (those presented in Section IV) and performance and energy characteristics of Gainestown with different LLC technologies. We "learn" this relationship for both the *fixed-capacity* and *fixed-area* configurations. Figure 3 shows this framework. For each workload, we use an array

	H_{r_g}	H_{r_l}	H_{w_g}	H_{w_l}	r_{uniq} (10^6)	w_{uniq} (10^6)	$90\%ft_r$ (10^3)	$90\%ft_w$ (10^3)	r_{total} (10^9)	w_{total} (10^9)
bzip2	18.03	10.23	11.72	5.90	5.99	5.88	2505.38	750.86	4.30	1.47
GemsFDTD	19.92	13.62	22.27	14.99	116.88	143.63	76576.59	113183.50	1.30	0.70
tonto	10.97	5.15	10.25	3.72	0.30	0.29	5.59	1.74	1.10	0.47
leela	10.13	4.07	8.95	3.01	2.26	5.06	1.59	1.29	6.01	2.35
exchange2	8.79	3.52	8.61	3.47	0.03	0.02	0.64	0.58	62.28	42.89
deepsjeng	11.31	5.69	11.86	5.93	58.89	68.28	4.79	4.33	9.36	4.43
vips	15.17	10.26	17.79	11.61	12.02	6.32	1107.19	1325.34	1.91	0.68
x264	16.14	7.43	11.84	4.04	11.40	9.28	1585.49	3.56	18.07	2.84
cg	19.01	11.71	18.88	11.96	2.30	2.36	1015.43	819.15	0.73	0.04
ep	8.00	4.81	8.05	4.74	0.563	1.47	0.84	113.18	1.25	0.54
ft	16.47	9.93	17.07	10.28	2.73	2.72	342.64	611.66	0.28	0.27
is	15.23	8.96	15.65	8.69	2.20	2.19	1228.86	794.26	0.12	0.06
lu	9.57	6.01	16.02	9.63	0.844	0.84	289.46	259.75	17.84	3.99
mg	17.97	11.80	16.93	10.18	7.20	7.29	4249.78	4767.97	0.76	0.16
sp	18.69	12.02	18.21	11.35	1.14	1.28	556.75	256.73	9.23	4.12
ua	13.95	8.17	11.23	5.69	1.32	1.57	362.45	106.25	9.97	5.85

TABLE VI: Workload Features. Heatmap on per-column basis showing extrema. H_{r_g}=global read entropy, H_{r_l}=local read entropy, H_{w_g}=global write entropy, H_{w_l}=local write entropy, r_{unique}=number of unique reads, w_{unique}=number of unique writes, $90\%ft_r$=90% read footprint, $90\%ft_w$=90% write footprint, r_{total}=total number of reads, w_{total}=total number of writes

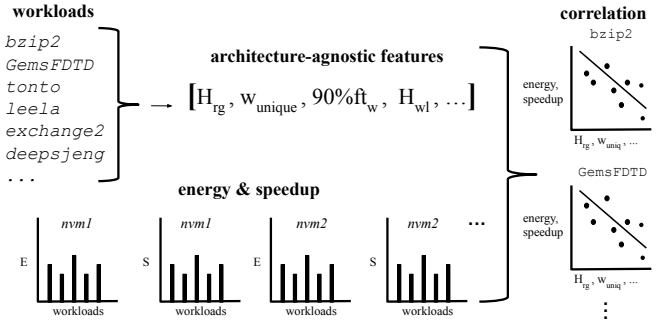
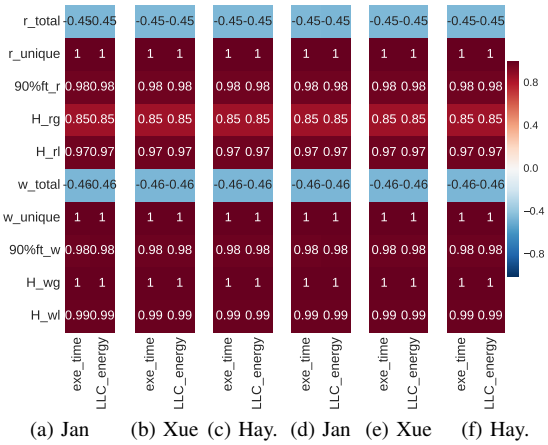


Fig. 3: Workload Characterization Framework



(a) Jan (b) Xue (c) Hay. (d) Jan (e) Xue (f) Hay.

Fig. 4: Feature correlation with energy and speedup for AI benchmarks with *fixed-capacity* (a)-(c) and *fixed-area* (d)-(f) LLC

of architecture-agnostic features (those listed in Table VI). This feature array along with the energy and speedup data from Section V is compiled. We compute linear correlation between the energy and speedup characteristics of an NVM and the feature array to learn which features are most useful in predicting performance and energy. We limit our analysis to *Jan$_S$*, *Xue$_S$*, and *Hayakawa$_R$* as these are the best performing and most energy efficient NVM-based LLCs.

Prior art simply states that NVM performance and energy is based on the number of reads and writes because of read and write asymmetry in NVMs. We use the workload metrics described in section IV-B for *all* workloads along with the performance and energy data for each NVM-based LLC configuration in Table III in our framework. For both fixed-area and fixed-capacity, LLC energy and system execution time is most highly correlated with the total number of reads and total number of writes. Therefore, in the case of a general purpose system, read and write footprint is indeed an appropriate metric for selecting NVM technology in the LLC. Though as current trends in computing shift from general purpose hardware to application-specific hardware, this assumption is stale.

Given that use cases are shifting more and more towards statistical inference, we use our framework just on the AI use cases from cpu2017 to learn the relationship between architecture-agnostic use case behavior and NVM. This is to emulate the process of selecting an LLC technology for a theoretical, modern domain specific archecture for statistical inference. Figures 4a–4f show the correlation results for *Jan$_S$-fixed-capacity*, *Xue$_S$-fixed-capacity*, *Hayakawa$_S$-fixed-capacity*, *Jan$_S$-fixed-area*, *Xue$_S$-fixed-area*, and *Hayakawa$_R$-fixed-area*, respectively. The higher the value in a cell, the higher the linear correlation between the corresponding metrics on the horizontal and vertical axes. For these benchmarks from the AI domain, performance and energy is most highly correlated with entropy, number of unique accesses, and 90% unique accesses while total number of reads and writes has almost no correlation with performance and energy.

We observe that the larger working sets of emerging workloads will be a greater predictor of the energy of their NVM-

based LLC and total system performance than the total number of writes. Therefore, in the case of the aforementioned hypothetical statistical inference specific architecture, the designer should consider picking an NVM device where density was the design target to accommodate the working set size.

VII. Future Work

As is the case when dealing with any emerging technology, there are abundant avenues for future investigation. In this section we discuss areas of future investigation related to the work presented here.

Apples-to-apples comparisons across NVM-based LLC works is critical for converging on accurate results. To this end, future work will continue to refine NVM-based LLC modeling methodologies. In addition to refining and developing better modeling heuristics, we will perform hardware validation of the models and characterize model error.

Despite recent efforts in mitigating the poor lifetime of the NVM classes studied in this work, it remains a major drawback and important consideration when selecting an NVM technology. Future work will characterize the extent to which architecture-agnostic features (like the ones studied in this work) will affect the lifetime of different NVMs.

As chips are becoming increasingly specialized, domain-specific architectures are a major factor in use case performance and power. Future work will look at more specialized architectures, such as GPUs, DSPs, and stencil processors using an adaptation of our feature correlation framework to study how the unique characteristics of an architecture affects selection of memory technology.

VIII. Conclusion

With the advent of NVM literature targeting the LLC, it is imperative that researchers are making apples-to-apples comparisons between different technologies. This work provides a unified modeling framework that researchers can use to compare NVMs, and we release a set of freely available NVM models on the web. Furthermore, an understanding of the interplay between NVM characteristics and workload behavior will enable researchers and industry professionals to select an NVM which will perform best for their use cases. We present a framework for analyzing how workload behavior affects LLC energy and system performance for different NVMs compared to an SRAM-based LLC baseline. It is shown that, for different use cases, different NVMs are more viable than SRAM for LLC adoption.

IX. Acknowledgements

The authors thank our colleagues in the Tufts Computer Architecture Lab, David Werner and Cesar Gomes, for their support in "taming" the vast amount of data. We would also like to thank Sparsh Mittal (IIT Hyderabad) for his guidance on NVM-based LLC modeling. Finally, we would like to thank the anonymous reviewers for their rigorous comments.

References

[1] J. Gaur, M. Chaudhuri, and S. Subramoney, "Bypass and insertion algorithms for exclusive last-level caches," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, June 2011, pp. 81–92.

[2] M. Chang, P. Rosenfeld, S. Lu, and B. Jacob, "Technology comparison for large last-level caches (l3cs): Low-leakage sram, low write-energy stt-ram, and refresh-optimized edram," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2013, pp. 143–154.

[3] S. M. Khan, Y. Tian, and D. A. Jimenez, "Sampling dead block prediction for last-level caches," in *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, Dec 2010, pp. 175–186.

[4] M. Chaudhuri, J. Gaur, N. Bashyam, S. Subramoney, and J. Nuzman, "Introducing hierarchy-awareness in replacement and bypass algorithms for last-level caches," in *2012 21st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Sep. 2012, pp. 293–304.

[5] A. Jaleel, W. Hasenplaugh, M. Qureshi, J. Sebot, S. Steely, and J. Emer, "Adaptive insertion policies for managing shared caches," in *2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Oct 2008, pp. 208–219.

[6] K. Korgaonkar, I. Bhati, H. Liu, J. Gaur, S. Manipatruni, S. Subramoney, T. Karnik, S. Swanson, I. Young, and H. Wang, "Density tradeoffs of non-volatile memory as a replacement for sram based last level cache," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ser. ISCA '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 315–327. [Online]. Available: https://doi.org/10.1109/ISCA.2018.00035

[7] Z. Wang, D. A. Jimnez, C. Xu, G. Sun, and Y. Xie, "Adaptive placement and migration policy for an stt-ram-based hybrid cache," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2014, pp. 13–24.

[8] H.-Y. Cheng, J. Zhao, J. Sampson, M. J. Irwin, A. Jaleel, Y. Lu, and Y. Xie, "Lap: Loop-block aware inclusion properties for energy-efficient asymmetric last level caches," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 103–114. [Online]. Available: https://doi.org/10.1109/ISCA.2016.19

[9] J. Ahn, S. Yoo, and K. Choi, "Dasca: Dead write prediction assisted stt-ram cache architecture," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2014, pp. 25–36.

[10] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3d stacked mram l2 cache for cmps," 03 2009, pp. 239–249.

[11] J. Wang, X. Dong, and Y. Xie, "Oap: An obstruction-aware cache management policy for stt-ram last-level caches," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2013, pp. 847–852.

[12] D. Kang, S. Baek, J. Choi, D. Lee, S. H. Noh, and O. Mutlu, "Amnesic cache management for non-volatile memory," in *2015 31st Symposium on Mass Storage Systems and Technologies (MSST)*, May 2015, pp. 1–13.

[13] L. Zhang, B. Neely, D. Franklin, D. Strukov, Y. Xie, and F. T. Chong, "Mellow writes: Extending lifetime in resistive memories through selective slow write backs," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 519–531.

[14] S. Manipatruni, D. E. Nikonov, and I. A. Young, "Energy-delay performance of giant spin hall effect switching for dense magnetic memory," *Applied Physics Express*, vol. 7, no. 10, p. 103001, sep 2014.

[15] J. Kim, B. Tuohy, C. Ma, W. H. Choi, I. Ahmed, D. Lilja, and C. H. Kim, "Spin-hall effect mram based cache memory: A feasibility study," in *2015 73rd Annual Device Research Conference (DRC)*, June 2015, pp. 117–118.

[16] M. Shihab, J. Zhang, S. Gao, J. Sloan, and M. Jung, "Couture: Tailoring stt-mram for persistent main memory," in *4th Workshop on Interactions of NVM/Flash with Operating Systems and Workloads (INFLOW 16)*. Savannah, GA: USENIX Association, 2016. [Online]. Available: https://www.usenix.org/conference/inflow16/workshop-program/presentation/couture-tailoring-stt-mram-persistent-main-memory

[17] H. Noguchi, K. Ikegami, K. Kushida, K. Abe, S. Itai, S. Takaya, N. Shimomura, J. Ito, A. Kawasumi, H. Hara, and S. Fujita, "7.5 a 3.3ns-access-time 71.2uw/mhz 1mb embedded stt-mram using physically

eliminated read-disturb scheme and normally-off memory architecture," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, Feb 2015, pp. 1–3.

[18] H. Noguchi, K. Ikegami, S. Takaya, E. Arima, K. Kushida, A. Kawasumi, H. Hara, K. Abe, N. Shimomura, J. Ito, S. Fujita, T. Nakada, and H. Nakamura, "7.2 4mb stt-mram-based cache with memory-access-aware power optimization and write-verify-write / read-modify-write scheme," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, Jan 2016, pp. 132–133.

[19] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "Energy reduction for stt-ram using early write termination," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, Nov 2009, pp. 264–268.

[20] S. Mittal, J. S. Vetter, and D. Li, "Writesmoothing: Improving lifetime of non-volatile caches using intra-set wear-leveling," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, ser. GLSVLSI '14. New York, NY, USA: ACM, 2014, pp. 139–144. [Online]. Available: http://doi.acm.org/10.1145/2591513.2591525

[21] S. Mittal and J. Vetter, "A technique for improving lifetime of non-volatile caches using write-minimization," *Journal of Low Power Electronics and Applications*, vol. 6, no. 1, 2016. [Online]. Available: https://www.mdpi.com/2079-9268/6/1/1

[22] A. Kawahara, K. Kawai, Y. Ikeda, Y. Katoh, R. Azuma, Y. Yoshimoto, K. Tanabe, Z. Wei, T. Ninomiya, K. Katayama, R. Yasuhara, S. Muraoka, A. Himeno, N. Yoshikawa, H. Murase, K. Shimakawa, T. Takagi, T. Mikawa, and K. Aono, "Filament scaling forming technique and level-verify-write scheme with endurance over 107 cycles in reram," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, Feb 2013, pp. 220–221.

[23] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, Feb 2011, pp. 50–61.

[24] Y. S. Shao and D. Brooks, "Isa-independent workload characterization and its implications for specialized architectures," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, April 2013, pp. 245–255.

[25] R. Wang, L. Jiang, Y. Zhang, and J. Yang, "Sd-pcm: Constructing reliable super dense phase change memory under write disturbance," *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '15)*, pp. 19–31, 2015.

[26] J. Fan, S. Jiang, J. Shu, Y. Zhang, and W. Zhen, "Aegis: partitioning data block for efficient recovery of stuck-at-faults in phase change memory," 2013, pp. 433–444.

[27] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie, "Overcoming the challenges of crossbar resistive memory architectures," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2015, pp. 476–488.

[28] H. Oh, B. Cho, W. Y. Cho, S. Kang, B. Choi, H. Kim, K. Kim, D. Kim, C. Kwak, H. Byun, G. Jeong, H. Jeong, and K. Kim, "Enhanced write performance of a 64 mb phase-change random access memory," in *ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference, 2005.*, Feb 2005, pp. 48–584 Vol. 1.

[29] Y. C. Chen, C. T. Rettner, S. Raoux, G. W. Burr, S. H. Chen, R. M. Shelby, M. Salinga, W. P. Risk, T. D. Happ, G. M. McClelland, M. Breitwisch, A. Schrott, J. B. Philipp, M. H. Lee, R. Cheek, T. Nirschl, M. Lamorey, C. F. Chen, E. Joseph, S. Zaidi, B. Yee, H. L. Lung, R. Bergmann, and C. Lam, "Ultra-thin phase-change bridge memory device using gesb," in *2006 International Electron Devices Meeting*, Dec 2006, pp. 1–4.

[30] S. K. et al, "A 0.1um 1.8v 256mb 66mhz synchronous burst pram," *International Solid-State Circuits Conference*, 2006.

[31] G. F. Close, U. Frey, J. Morrish, R. Jordan, S. C. Lewis, T. Maffitt, M. J. BrightSky, C. Hagleitner, C. H. Lam, and E. Eleftheriou, "A 256-mcell phase-change memory chip operating at 2+ bit/cell," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 6, pp. 1521–1533, June 2013.

[32] S. Chung, K. M. Rho, S. D. Kim, H. J. Suh, D. J. Kim, H. J. Kim, S. H. Lee, J. H. Park, H. M. Hwang, S. M. Hwang, J. Y. Lee, Y. B. An, J. U. Yi, Y. H. Seo, D. H. Jung, M. S. Lee, S. H. Cho, J. N. Kim, G. J. Park, G. Jin, A. Driskill-Smith, V. Nikitin, A. Ong, X. Tang, Y. Kim, J. S. Rho, S. K. Park, S. W. Chung, J. G. Jeong, and S. J. Hong, "Fully integrated 54nm stt-ram with the smallest bit cell dimension for high density memory application," in *2010 International Electron Devices Meeting*, Dec 2010, pp. 12.7.1–12.7.4.

[33] G. Jan, L. Thomas, S. Le, Y. J. Lee, H. Liu, J. Zhu, R. Y. Tong, K. Pi, Y. J. Wang, D. Shen, R. He, J. Haq, J. Teng, V. Lam, K. Huang, T. Zhong, T. Torng, and P. K. Wang, "Demonstration of fully functional 8mb perpendicular stt-mram chips with sub-5ns writing for non-volatile embedded memories," in *2014 Symposium on VLSI Technology (VLSI-Technology): Digest of Technical Papers*, June 2014, pp. 1–2.

[34] Y. Umeki, K. Yanagida, S. Yoshimoto, S. Izumi, M. Yoshimoto, H. Kawaguchi, K. Tsunoda, and T. Sugii, "A negative-resistance sense amplifier for low-voltage operating stt-mram," in *The 20th Asia and South Pacific Design Automation Conference*, Jan 2015, pp. 8–9.

[35] L. Xue, Y. Cheng, J. Yang, P. Wang, and Y. Xie, "Odesy: A novel 3t-3mtj cell design with optimized area density, scalability and latency," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2016, pp. 1–8.

[36] Y. Hayakawa, A. Himeno, R. Yasuhara, W. Boullart, E. Vecchio, T. Vandeweyer, T. Witters, D. Crotti, M. Jurczak, S. Fujii, S. Ito, Y. Kawashima, Y. Ikeda, A. Kawahara, K. Kawai, Z. Wei, S. Muraoka, K. Shimakawa, T. Mikawa, and S. Yoneda, "Highly reliable taox reram with centralized filament for 28-nm embedded application," in *2015 Symposium on VLSI Technology (VLSI Technology)*, June 2015, pp. T14–T15.

[37] P. M. Palangappa and K. Mohanram, "Compex: Compression-expansion coding for energy, latency, and lifetime improvements in mlc/tlc nvm," *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 90–101, 2016.

[38] S. Mittal, J. S. Vetter, and D. Li, "Lastingnvcache: A technique for improving the lifetime of non-volatile caches," in *2014 IEEE Computer Society Annual Symposium on VLSI*, July 2014, pp. 534–540.

[39] S. Mittal and J. S. Vetter, "Equalwrites: Reducing intra-set write variations for enhancing lifetime of non-volatile caches," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 1, pp. 103–114, Jan 2016.

[40] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, July 2012.

[41] M. Poremba and Y. Xie, "Nvmain: An architectural-level main memory simulator for emerging non-volatile memories," in *2012 IEEE Computer Society Annual Symposium on VLSI*, Aug 2012, pp. 392–397.

[42] M. Poremba, S. Mittal, D. Li, J. S. Vetter, and Y. Xie, "Destiny: A tool for modeling emerging 3d nvm and edram caches," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2015, pp. 1543–1546.

[43] T. E. Carlson, W. Heirmant, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2011, pp. 1–12.

[44] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006. [Online]. Available: http://doi.acm.org/10.1145/1186736.1186737

[45] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, January 2011.

[46] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga, "The nas parallel benchmarks," *Int. J. High Perform. Comput. Appl.*, vol. 5, no. 3, pp. 63–73, Sep. 1991. [Online]. Available: http://dx.doi.org/10.1177/109434209100500306

[47] R. Adolf, S. Rama, B. Reagen, G. Wei, and D. M. Brooks, "Fathom: Reference workloads for modern deep learning methods," *CoRR*, vol. abs/1608.06581, 2016. [Online]. Available: http://arxiv.org/abs/1608.06581

[48] H. Zhu, M. Akrout, B. Zheng, A. Pelegris, A. Jayarajan, A. Phanishayee, B. Schroeder, and G. Pekhimenko, "Benchmarking and analyzing deep neural network training," in *2018 IEEE International Symposium on Workload Characterization (IISWC)*, Sep. 2018, pp. 88–100.

[49] M. Lapuerta, "The parsec benchmark suite," Jan 2012.

[50] M. Lui, K. Sangaiah, M. Hempstead, and B. Taskin, "Towards cross-framework workload analysis via flexible event-driven interfaces," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, April 2018, pp. 169–178.