Leveraging Data Compression for Performance-Efficient and Long-Lasting NVM-based Last-Level Caches

C. Escuin*, A.A. Khan†, P. Ibáñez*, T. Monreal‡, D. Navarro*, J.M. Llabería‡, J. Castrillon†, V. Viñals*
*Universidad de Zaragoza (Zaragoza, Spain), †TU Dresden (Dresden, Germany), ‡Universitat Politècnica de Catalunya (Barcelona, Spain)

I. Introduction

The goal of the last-level cache (LLC) in the memory hierarchy of a multiprocessor system is to filter main memory requests and deliver them faster to the private levels. The number of integrated cores on a chip is increasing and so it is the LLC size. Most of LLCs are built using SRAM technology that does not scale well in terms of density and static power. Given this limitation, NVMs are interesting alternatives to replace or augment on-chip SRAM LLC due to their higher density and lower static power [2], [6], [8]. However, NVM write operations are costly in terms of latency and energy, and gradually wear out the bitcells until they become defective. Compared to SRAM, NVM endurance is limited and usually modeled by a normal distribution of mean 10^k write operations, k depending on the technology, manufacturer and target market [6], [7].

Several works address the write endurance problem in complementary ways such as evenly distributing writes across the whole memory structure (wear-leveling [6]), reducing the number of writes to the NVM [2], [8], or using redundant storage [7]. In order to cope with hard faults, memory structures must be provided with error correction codes, ECCs. In addition, when a hard fault occurs (a bitcell fault), it is required to deactivate the corresponding memory region to ensure the correct system operation. Traditionally, the whole cache frame is disabled when a bitcell fails [1].

In order to push the state-of-the-art in NVM-LLCs, it is needed to 1) reduce the number of writes, 2) maintain uniform wear on all bit cells, and 3) continue to operate even if some of the rated capacity is out of service. Our **first contribution is L2C2 [5]**: a novel fault-tolerant NVM-LLC that combines byte-level disabling, data compression, and an intra-frame wear-leveling mechanism. Data compression, apart from reducing the bytes written (to increase NVM lifetime), enables partially degraded frames to allocate compressed blocks (to increase hit rate). To this end, L2C2 maintains a fault map that points at the faulty bytes and a novel circuitry that evenly distributes the bytes of the incoming compressed block among the non-faulty bytes of the target frame.

NVM-SRAM hybrid LLCs combine the best of both worlds, i.e. high capacity and long lifetime. The NVM part provides capacity, thanks to its high density, while the SRAM part provides endurance, absorbing write-intensive flows. Naively filling blocks either in the NVM or SRAM part leads to an early wear out of the NVM bitcells. Thus, the lifetime improvement of state-of-the-art proposals [2] is achieved by conservatively inserting read-intensive blocks in the NVM part, which in turn limits the hybrid LLC performance. Besides, previous works do not consider compressing the blocks or using partially defective frames. Our **second contribution [4]** leverages L2C2 microarchitecture to propose new hybrid LLC insertions policies that optimize for both performance and lifetime. These policies consider the compression size, the read-reuse, and the write-reuse of the incoming blocks when deciding where to place them. Besides, we achieve runtime adaptivity through a threshold-based mechanism that is able to further balance the performance-lifetime tradeoff.

Previous works indirectly evaluate lifetime improvement by means of the relative reduction in the number of writes [8], or by using approximate aging models in the context of NVM main memory [7]. To the best of our knowledge, there is no model able to forecast the performance over time of these NVM-LLCs whose capacity drops over time due to write operations. Therefore, our **third contribution [5]** is such a forecasting procedure, suitable for multicore NVM-LLCs. It combines simulation-prediction epochs. Within each epoch, a cycle-accurate simulation and successive byte-failure predictions are performed.

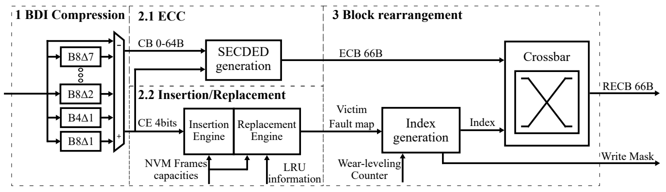
II. L2C2: Byte-level Fault-tolerant NVM-LLC

We assume a SECDED mechanism (note that ECCs are already present in commercial SRAM LLCs), able to detect, correct, and identify the byte the hard fault is suffered by. We propose using a *fault map* that records the health of each byte, distinguishing between operational and failed, incurring 1/8 overhead with respect to the capacity of the NVM data array. We design a logic circuit that, based on this information and, leveraging data compression, can evenly distribute the bytes of the compressed block among the non-faulty bytes of a partially degraded frame. The circuits of both reading and writing a block are analogous. Below, we explain the flow of writing a block while the reading one is detailed in the original paper [5].

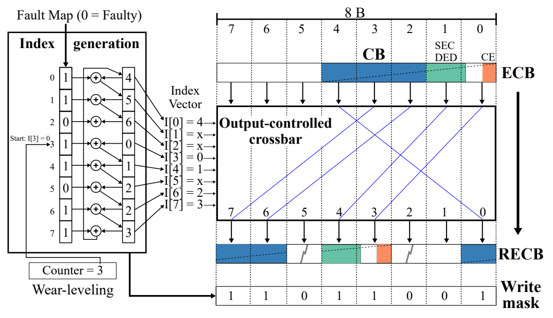
Figure 1a depicts the flow of filling a block into L2C2. First, the compressor generates the compressed block (CB). Next, based on CB the ECC bits are computed conforming ECB and, in parallel to this, the insertion/replacement mechanism selects the target frame among the frames whose capacity (regarding the fault map) is large enough to host ECB. Finally, the intra-frame wear-leveling mechanism by means of a global counter (common to all frames) points at the initial write position of ECB within the frame. Regarding this counter and the fault map of the target frame, the ECB bytes are scattered (RECB) among the non-faulty bytes of the frame. Figure 1b illustrates how the output-controlled crossbar redirects the ECB bytes by generating an index vector that controls the crossbar. For instance, I[3]=0 means that byte 0 of ECB is placed in byte 3 of RECB, which is the first byte pointed at by the wear-leveling counter. Using VLSI synthesis, the block rearrangement circuitry showed to be feasible for both reading and writing in terms of latency, area and power consumption [5].

III. Hybrid LLCs Insertion Policies

In this work, we focus on bridging the performance-lifetime disparities from state-of-the-art hybrid LLCs leveraging L2C2 microarchitecture to extend the NVM part, see Section II. To do so, we classify the incoming blocks to the hybrid LLC into three categories based on their reuse properties: read-reuse, write-reuse, and non-reuse. Read-reuse blocks are the clean blocks that have been previously hit in the LLC, while write-reuse blocks are the dirty ones. Read-reuse ones are candidates to stay long time in the LLC so will be placed in the NVM part while write-reuse ones will produce consecutive write operations and will thereby steered towards the SRAM part. Non-reuse blocks,



(a) Block writing flow.



(b) Example of a 5-byte block rearrangement for writing.

Fig. 1. Writing a block flow (a) and 5-byte block rearrangement example (b).

which are the ones that have not shown any reuse yet, are further classified in small and big blocks if their compressed size is lower or greater than the compression threshold (CP_{th}), respectively [4].

This CP_{th} controls the write traffic to both NVM and SRAM parts: the higher the CP_{th} the greater the number of non-reuse blocks that are mapped to NVM. Not using the optimal CP_{th} drives to performance drop due to an unbalanced number of references to both parts. CP_{th} showed to need runtime adaptivity since the optimal performance was achieved by different CP_{th} depending on the hybrid LLC effective capacities, the execution phase, and the workload. To do so, we propose CP_SD, an insertion policy that address this runtime adaptivity with Set Dueling. During execution epochs of 2 Mcycles, different groups of cache-sets employ different fixed CP_{th} and the remaining sets follow the group of sets whose CP_{th} brought the maximum number of hits in the last epoch. Moreover, we also propose a rule-based mechanism to further tune the write-traffic to the NVM part allowing us to trade performance in exchange of lifetime.

IV. Forecasting procedure

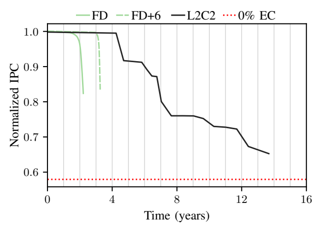
From the methodology perspective, in order to forecast the performance of these NVM-LLCs over time, we could first initialize the remaining number writes (RW) every byte can support regarding an endurance model of the NVM technology [6], [7]. A naive, but exact, approach would simulate in detail the progressive degradation of the NVM-LLC reflecting in RW every time a write operation is performed. When a byte reaches its maximum number of writes, it is disabled and the simulation continues with the system a little bit more degraded. However, to get realistic results, the simulation must be cycle-accurate and driven by a realistic workload, but in doing so we could only forecast time lapses of few milliseconds.

In order to reduce this simulation time, we propose an approximate forecasting procedure that combines simulation-prediction epochs [5]. In every epoch, a cycle-accurate simulation receives the NVM-LLC state (regarding the worn out regions) and returns the *write rate to NVM frames*. The prediction phase makes K consecutive predictions (the next K bytes that become faulty) from a single simulation. To do so, we first compute the *average write rate* of the frames that are *similar*. This means aging with the same average write rate to all the frames that fulfill two conditions: 1) frames that have a similar effective capacity and thereby have the same capability to allocate compressed blocks, and 2) frames that belong to sets with the same health state (capability to allocate compressed blocks that has each frame within a set).

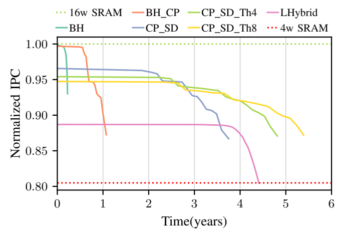
V. Evaluation: Performance vs. Lifetime

Our system features a 4-core multiprocessor with the latency parameters of a 16MB LLC. We use gem5 and HyCSim, which is a tool for design space exploration for hybrid LLCs [3]. The workload consists of application mixes taken from SPEC CPU 2006 and 2017. The original papers offer further configuration details [4], [5].

Figure 2a shows the performance evolution until the NVM-LLC effective capacity drops 50% of its nominal one. 0% EC represents a fully impaired NVM-LLC. L2C2 is compared against two baseline systems: frame disabling (FD) that does not employ data compression and thereby disables the whole frame when a byte becomes faulty [1] and FD+6 that extend FD functionality providing every frame with 6 ECPs [7]. Compared to FD, FD+6 and L2C2 improves lifetime by 1.5× and 6.2×, respectively. L2C2 gracefully loses capacity postponing the gradual performance loss.



(a) Monolithic NVM



(b) Hybrid LLC

Fig. 2. Performance evolution until the NVM effective capacity drops 50%.

Figure 2b shows the performance evolution of a hybrid LLC over time, until the capacity of the NVM part drops to 50%. Twelve and four ways have been devoted to NVM and SRAM parts, respectively. Bounds of SRAM-only LLCs are also plotted. The baseline configuration (BH) employs frame disabling and the insertion policy fills the block to the LRU way, regardless of its technology. Naively writing blocks to the NVM part leads to 50% of its capacity being exhausted in less than three months. Compared to BH, LHybrid [2] extended with frame disabling improves LLC lifetime by more than 19× at the cost of significant performance drop (11%), due to its conservative insertion policy. CP_SD outperforms LHybrid by 9% while achieving a comparative lifetime. The rule-based mechanism shows that by compromising, for instance, 1.1% (CP_SD_Th4) and 1.9% (CP_SD_Th8) performance, the NVM lifetime can be further increased by 28% and 44%, respectively.

References

[1] J. Chang *et al.*, "The 65-nm 16-mb shared on-die l3 cache for the dual-core intel xeon processor 7100 series," *JSSC*, 2007.
[2] H.-Y. Cheng *et al.*, "Lap: Loop-block aware inclusion properties for energy-efficient asymmetric last level caches," in *ISCA*, 2016.
[3] C. Escuin *et al.*, "Hycsim: A rapid design space exploration tool for emerging hybrid last-level caches," in *RAPIDO'22*, 2022.
[4] C. Escuin *et al.*, "Compression-aware and performance-efficient insertion policies for long-lasting hybrid llcs," in *HPCA*, 2023. [Online]. Available: https://www.dropbox.com/s/vvz916qvgj2jrv3/hpca2023-publicada.pdf
[5] C. Escuin *et al.*, "L2c2: Last-level compressed-contents non-volatile cache and a procedure to forecast performance and lifetime," *PLOS ONE*, 2023. [Online]. Available: https://doi.org/10.1371/journal.pone.0278346
[6] H. Farbeh *et al.*, "Floating-ecc: Dynamic repositioning of error correcting code bits for extending the lifetime of stt-ram caches," *TonC*, 2016.
[7] S. Schechter *et al.*, "Use ecp, not ecc, for hard failures in resistive memories," *ACM SIGARCH Computer Architecture News*, 2010.
[8] S. Seyedzadeh *et al.*, "Enabling fine-grain restricted coset coding through word-level compression for pcm," in *HPCA*, 2018.