Literature Review

Submitted by -
Inayat Kaur (2020csb1088)
Pravesh Sanjay Jamgade (2022CSZ0007)

Outline

Introduction

Until recently, SRAM has been a prominent choice of use in cache hierarchy. Recent growth in increase in chip multiprocessor has an impact on increase in bandwidth demand. SRAM being volatile memory requires supply of power to keep data in cells active. This power requirement increases significantly with increase in SRAM size. Hence just increasing size to accommodate for large bandwidth requirements is a costly solution. Another alternative to SRAM is eDRAM, but the same argument can be applied here is the power supply it requires for periodic refresh. Although eDRAM is less expensive, provides higher density than SRAM but energy consumption overshadows these advantages.
More recently, non-volatile memories have been gaining popularity. Primary reason being similar read energy and latency as of SRAM. STT-RAM is one of the NVM technologies among others such as PCRAM, ReRAM . STT-RAM are much denser than SRAM and there is negligible static (leakage) power dissipation. However, write operations to STT-RAM require significantly higher write energy and longer write latency. The long write latency degrades performance by causing write induced interference to subsequent read operations. Also there is another problem of

lifetime of NVM cells, which is project to be $\sim 10^{15}$, but a developed prototype has given this value is around 4×10^{12}. The lifetime of SRAM and DRAM is more than 10^{15}. Hence NVM has high chances of getting damaged if write operations are performed repetitively to a single location.

Recent body of work has been working on reducing energy consumption, improving lifetime and performance of NVM.

Background

Sparsh and Jeffrey (2016)[1] proposed ENLIVE which is a write minimization technique; they have used a SRAM buffer as HotStore to place write-intensive blocks and reduce memory pressure. They maintained a counter per block to measure write intensity on the block. This has energy as well as area overhead. From each set only one block is allowed to be put into HotStore. There is no separate tag directory for data placed in HotStore, they use the L2 tag directory for HotStore data to reduce search time. ENLIVE provides 8.47x, 14.67x and 15.79x improvement in lifetime for two, four and eight core systems respectively. They compared ENLIVE with LastingNVCache and PoLF and it has performed better on performance, relative lifetime and reduction is loss of energy. LastingNVCache flush the write intensive lines and upon again a read miss allocates cache line to a cold block. This is an intra-set write variation technique. PoLF is a probabilistic line flush, it probabilistically flushes lines when a global counter saturates. Both methods increase off-chip memory accesses and do not guarantee reduced LLC write pressure instead it will more likely increase it. The advantage of ENLIVE over compared state-of-art is using SRAM buffer to absorb write intensive cache lines instead of invalidating them. But the overhead is as we said counters per block (to measure write intensity) and only one block per set capacity in SRAM buffer.

Wang et al., (2014), performed experimental studies to show the types of writes access patterns at LLC. In their studies they pointed to core side writes (core writes) and memory side writes (demand and prefetch write). They classified these writes further to show what fraction of them are more likely to dead-on-arrival and whether they have re-reference like in burst or distant. Based on these classifications, they proposed data placement techniques. They proposed a hybrid cache management policy, SRAM portion to absorb writes such as dead-on-arrival, burst writes, bypassing LLC for filling writes. Their main goal is to reduce power consumption and improve performance. Proposed solution is reducing LLC write pressure and that affected reduction in power consumption, interference due to writes is reduced which is responsible for long write latency hence performance improved. Take away is write pressure is already reduced but write variation still will still cause decreased lifetime of STT-RAM portion.

Hsiang-Yun et al., (2016) proposed an inclusion policy. They performed a set of studies on existing inclusion policies such as non-inclusion and exclusion and found out characteristics from both of them that could be helpful in reducing writes and increasing capacity of LLC. Their goal is to reduce writes which will reduce power consumption and improve latency much like Wang et al., (2016), targeted for. Loop blocks are read blocks often evicted from L2 and again reused, hence they jump on and off between L2 and LLC. In their analysis, they observed blocks doing clean trips between L2 and LLC. Blocks are being written redundantly at L2 and LLC, evicted blocks from L2 are either dropped if they are clean or they are write-back to LLC if they are dirty (in case of inclusive caches). Blocks are inserted in L2 upon fill from memory, and every block from L2 eviction is written to LLC which causes extra writes (in case of exclusive caches). They designed a policy to write loop blocks to STT-RAM and non-loop blocks to SRAM. There is a replacement policy at SRAM and STT-RAM which prefers invalid, non-loop and loop blocks for replacement candidates (decreasing priority). Take away is the proposed solution, actively looking at reducing writes because of unnecessary writes by loop blocks and non-loop blocks on STT-RAM.

Junwhan et al., (2016) proposed a policy to minimize writes at LLC based on studies where they have classified writes requests. They have identified three types of writes: dead-on-arrival, dead-value-fills, closing-writes. With the given classification they proposed a policy to make a decision whether to write or skip (bi-directional) LLC. The predictor they have used to classify such writes gives 91% accuracy with a false-positive rate of 1.4%. The main focus is to reduce write operations at LLC.

Kunal et al., (2018) showed how bypassing cache can give diminishing returns in performance improvement. Their observations is that the request queue occupying more number of writes can degrade performance in STT-RAM cache as the writes operations take long latency. And requests behind the write operation waiting in the queue will be delayed.

Challenge And Opportunities

The performance improvement is mainly because of intelligent bypassing as demonstrated by Kunal et al., (2018). But again these have limitations as more aggressive bypassing can lead to increased traffic at LLC and hence reduced performance. Hence categorizing writes based on their lifetime in cache and then applying bypassing could be an effective way to manipulate knobs on bypassing.

The methods proposed by Wang et al., (2014), Junwhan et al., (2016), Hsiang-Yun et al., (2016) are basically doing write minimization at LLC. This is why reduction in power consumption and improved performance can be seen in their results. All writes at LLC are not guaranteed to have reduced write variation, hence these approaches could not guarantee improved lifetime of STT-RAM. But their approach to absorb write intensive blocks could be orthogonal with techniques that can reduce write variation. Also, in these cases a good predictor design plays an important role, as more accuracy can reduce extra writes.

Goal

The goal is to design a STT-RAM LLC management policy to improve performance, reduce power consumption and improve lifetime. This can be achieved by reducing write traffic at LLC and for remaining writes we can do wear leveling to reduce write variation.

Approach

The write counters are used per set to monitor write accesses per set. When write counters values reach a certain threshold, then such sets can be used to bypass writes operations. Before bypassing writes, we can predict the type of write either dead or live, and based on that decide whether to bypass them to main memory or use a SRAM buffer to absorb them. We will need to design an accurate predictor with very few false positives.

Conclusion

Described recent work's are using write minimization techniques to show the reduction in number of writes, which implicitly improves performance and reduces energy consumption by STT-RAM. However, they do not mention any numbers about improvement in lifetime of STT-RAM based LLC. But since the number of writes are reduced, there has to be a significant impact on the lifetime of the NVM unless the write variation is less. We aim to understand the impact of bypassing dead blocks on the lifetime of cache and design the policy that will compliment energy reduction and performance improvement. We aim to develop a solution that maintains a balance among the three important parameters i.e. performance, energy and lifetime of NVM.

References

1. Sparsh Mittal, Jeffrey S. Vetter,"A Technique for Improving Lifetime of Non-Volatile Caches Using Write-Minimization", Journal of Low Power Electronics and Applications,10.3390/jlpea6010001, 2016.
2. Zhe Wang,Daniel A. Jimenez,Cong Xu,Guangyu Sun,Yuan Xie,"Adaptive Placement and Migration Policy for an STT-RAM Based Hybrid Cache", 10.1109/hpca.2014.6835933,2014.
3. Hsiang-Yun Cheng,Jishen Zhao,Jack Sampson,Mary Jane Irwin,Aamer Jaleel,Aamer Jaleel,Yu Lu,Yuan Xie,"LAP: Loop-Block Aware Inclusion Properties for Energy-Efficient Asymmetric Last Level Caches",10.1145/3007787.3001148,2016.
4. Junwhan Ahn,Sungjoo Yoo,Sungjoo Yoo,Sungjoo Yoo,Kiyoung Choi,"DASCA: Dead Write Prediction Assisted STT-RAM Cache Architecture",International Symposium on High-Performance Computer Architecture,10.1109/hpca.2014.6835944,2014.
5. Kunal Korgaonkar,Kunal Korgaonkar,Kunal Korgaonkar,Ishwar Bhati,Hongyu Liu,Huichu Liu,Jayesh Gaur,Sasikanth Manipatruni,Sreenivas Subramoney,Tanay Karnik,Steven Swanson,Ian Young,Ian A. Young,Hong Wang,"Density Tradeoffs of Non-Volatile Memory as a Replacement for SRAM based Last Level Cache",International Symposium on High-Performance Computer Architecture,10.1109/isca.2018.00035,2018.
6. S.Agarwal, H.Kapoor,"Improving the Lifetime of Non-Volatile Cache by Write Restriction",IEEE Transactions on Computers, 10.1109/TC.2019.2892424,2019.

Detailed summary about the above mentioned papers is given on the following pages.

Paper: A Technique for Improving Lifetime of Non-Volatile Caches Using Write-Minimization
Author: S. Mittal et. al
Venue: Journal of Low Power Electronics and Applications, 2016

Your summary should contain the following points:
1. Problem addressed in the paper
Improving lifetime of NVM based LLC cache.
2. Motivation for the problem/how relevant or critical is the problem addressed?
Write Variation across blocks for different workloads is different. This might lead to damage to a cell. There are existing methods that try to reduce write variation across cache such as LastingNVCache which periodically flushes the write intensive cache line and upon subsequent read miss fills it in the cold cache block (less write intensive cache block). Problem with such an approach is that it increases memory traffic and number of writes as well.

3. Motivation for the solution/insights that helped to develop the solution?
Write variation is due to few uneven writes across cache. Capture write intensive blocks on a small SRAM buffer can help reduce writes on the NVM cache.

4. Details of the solution proposed
a. Theory
ENLIVE is the proposed solution. It stands for **EN**hancing the **LI**fetime of non-volatile cach**E**s. It uses a SRAM buffer called HotStore. Its purpose is to migrate write intensive blocks to HotStore so that the future accesses also reduces the number of writes to the NVM cache which translates into enhanced cache lifetime.
HotStore only stores the data blocks and does not require a separate tag field. It uses tags from the L2 tag directory only, hence no full associative search is not required on HotStore. Only one block from a set allowed to store in HotStore. Insertion in HotStore is done on the basis of write counter value of cache block, if it is greater than threshold then it is chosen as candidate for promotion to HotStore. Only one block is chosen for promotion per cache set. Eviction from HotStore is done on the basis of number of writes, if the replacement candidate has fewer writes than incoming cache block. If a block from the same set is already in HotStore, then write counter values are compared and larger write counter value blocks are placed in HotStore. To select replacement candidates, HotStore uses Least Number of Writes and Not Recently used replacement policies.

i. Are there any heuristics?
Write intensity and a threshold to determine which block to be sent to HotStore.

b. Implementation:

i. Which simulator, which data set, state-of-the-art works for comparison, what

is the baseline.
Snipersim simulator.DESTINY for energy estimation.
Workloads, SPEC CPU 2006 suite, HPC.
Baseline is LastingNVCache and Probabilistic Line Flush (PoLF)

ii. What are the metrics of comparison? How to they measure them.
(A)Relative cache lifetime
(B)Weighted SpeedUp
(C)Percentage Energy Loss
(D)Absolute increase in MPKI

5. Your own critique (Hint: Think as if you are the official paper reviewer)
The SRAM part doesn't have its own tag array and to find the location of cache blocks in SRAM it uses counter values (set index and block number) from NVM cache blocks. It is anyway taking space for these counters, a cache block per counter value. Rather it could have been convenient to use parallel search to both NVM and SRAM. SRAM buffer has static power consumption. Only one block is allowed to reside in the SRAM buffer.

a. Comment on novelty of the solution

b. Your insight into the strengths and weakness of the proposed solution
i. Why, where this solution works.
SRAM consumes write intensive blocks.
ii. Where it does not work
Maybe when workloads are really write intensive and the SRAM buffer can't hold such blocks for longer.

c. Reasons on outliers (related to 5 b ii)

d. Any possible improvements
With heavy write intensive workloads, for SRAM it will not be possible to capture all such blocks eventually it will have to evict such blocks to NVM. A better replacement policy will be required. Better knowledge of best working or flexible threshold value for which writes can be redirected needs to be considered.

6. Regarding related works:
a. Think as the author and understand how the author would have picked the related works
i. How to decide whether to quantitatively or qualitatively compare?
b. Make note of the most influential related works, most recent impactful related works.

Paper: <u>Adaptive Placement and Migration Policy for an STT-RAM Based Hybrid Cache</u>
Author: Zhe Wang et. al

Venue: HPCA 2014

1. Problem addressed in the paper

Effectively managing hybrid STT-RAM and SRAM based LLC architecture to reduce power consumption and improve performance.

2. Motivation for the problem/how relevant or critical is the problem addressed?

Latency of read operation in STT-RAM is similar to SRAM but the same is not true for write operation. Write latency is more for STT-RAM, plus there is significant write variation which can damage STT-RAM cells. Previous works propose block placement and migration policies for write intensive blocks.

The write requests to LLC can happen because of demand requests missed using fill back requests from main memory, core write miss upon data hit in cache and prefetch write upon prefetch miss in cache. The analysis of write access patterns at LLC gives insight, as to which patterns most likely are dead on arrival, immediately reuse and distant reuse. Based on this analysis a policy could be designed to use Hybrid cache more effectively.

Write accesses are classified into 2 types

(A) Read Range, writes from demand fills and prefetch fills. It is the largest interval between consecutive reads of block from time it is filled into LLC until the time it is evicted.

(B) Depth Range, writes from core. It is the largest interval between write access and next core write accesses. Each LLC write request could be one of the demand write, prefetch write on core write.

Again each of these writes can be further classified as zero read/depth range, immediate read/depth range and distant read/depth range. Read range is used for prefetch and demand write, because when these blocks are written in LLC then usability of such blocks determine if at all any energy savings can be achieved or not. Similarly for Depth range, it is being used with core write requests.

From the analysis, following composition of access pattern is found:

	Zero	Immediate	Distant
Prefetch Write	26%	56.9%	17.5%
Demand Write	61.2%	Sums up to 38.8%	
Core Write	49%	32.9%	18%

3. Motivation for the solution/insights that helped to develop the solution?

Inaccurate prefetches are 26%, as they are never referenced apart from initial reference which itself is a prefetch write. Hence better to put them in SRAM.

Immediate bursts were 56.9% within cache set size. Hence better to place in SRAM.

And distant references are 17.5%, it would be better if they stay in cache for longer. Place them into SRAM and if they are still alive then migrate to STT-RAM.

Demand writes with zero reuse are 61.2%, it would be better to not write them, hence bypassing could be a better choice to be made here. Remaining useful reuse is 38.8%. Put them into STT-RAM to make use of STT-RAM capacity.

Core writes are zero reuse with 49% of occurrence. Keep them as it is in their cache lines. 32.9% are immediate, put them in SRAM and 18% are distant reuse.

4. Details of the solution proposed

a. Theory

Based on analysis of access patterns, policy is designed to make better placement decisions.

i. Are there any heuristics?

Prediction table used to make placement design based on type of pattern.

b. Implementation:

Access pattern predictor is composed of prediction table and pattern simulator.

Prediction is made for following cases:

(A) When a core write request hits in STT-RAM lines, the Write Burst Prediction Table (WPT) is accessed.

(B) When a read hit request in the SRAM lines, the Dead Block Prediction Table (DPT) is accessed.

(C) On Demand request, the Dead Block Prediction Table is accessed.

Prediction table structure is the same for both WPT and DPT.

WPT indexed by write pc and it has lru recency information. DPT indexed by read pc and it has lru recency information as well.

(tag, read PC, lru, valid) and (tag, write PC, lru, valid)

Updating prediction table:

(A) On Demand hit, DPT accessed by "read PC" to decrease counter value implying not-dead. On eviction the same counter is increased to imply the block is dead. LRU recency is updated for every demand request and prefetch write requests.

(B) "Write PC" accesses WPT and increments counter value when core-write request is hit implies it is write burst. And decrements when block is evicted, implies no write burst.

i. Which simulator, which data set, state-of-the-art works for comparison, what is the baseline.

MARSx86 + DRAMSim2 simulator.

Baseline micro-architecture:

4GHz, 1 core/ 4 core CMP, OoO, ROB=128, LD=48, ST=44, issue/decode width=4.

64 KB Cache block size.

L1-I : 64KB/2-way, private, LRU

L2: shared, LRU

DRAM: DDR3-1333, open-page, 2-channel, 8-bank/channel, FR_FCFS,

32-entry/channel write buffer

Workloads:

Mix workloads from SPEC CPU 2006 benchmarks.

ii. What are the metrics of comparison? How to they measure them.

(A) Reduced Writes and Endurance: Distribution of write operations to STT-RAM lines normalized to all write operations to LLC. 28.9% avg. write reduction.
(B) Performance: Speedup normalized to 8MB SRAM LLC. 14.8% avg. performance improvement.
(C) Power: Power normalized to 8MB SRAM LLC.
(D) Prediction: false rate between 2.1% to 14.8%, average coverage 71.7%
(E) Memory energy:
(F) Storage overhead

5. Your own critique (Hint: Think as if you are the official paper reviewer)

a. Comment on novelty of the solution

b. Your insight into the strengths and weakness of the proposed solution
Weakness: predictor based approach and it has a good amount of false positive rate.
i. Why, where this solution works.
ii. Where it does not work

c. Reasons on outliers (related to 5 b ii)
APM reduces cache pollution, reduces writes which in return improves performance, reducing power consumption.

d. Any possible improvements
6. Regarding related works:
a. Think as the author and understand how the author would have picked the related works
i. How to decide whether to quantitatively or qualitatively compare?
b. Make note of the most influential related works, most recent impactful related works

---

Paper: LAP: Loop-Block Aware Inclusion Properties for Energy-Efficient Asymmetric Last Level Caches
Author : Hsiang-Yun Cheng, Jishen Zhao, Jack Sampson, Mary Jane Irwin, Aamer Jaleel, Yu Lu, and Yuan Xie
Venue: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture
1. Problem addressed in the paper
The paper tries to minimize the energy consumption of NVM based LLC keeping in mind their read/write energy asymmetry.
2. Motivation for the problem

SRAM based cache consumes large amounts of static energy due to leakage current. Instead of SRAM, NVMs can be used which have larger density and low static energy. But they have large asymmetries in read and write operation energy requirements. Aso, dynamic write energy exceeds leakage energy in NVMs. Hence reducing the number of writes in NVMs can lead to energy reduction.

3. Motivation for the solution

Previous approaches try to reduce the number of writes or decrease the number of bits to be written per write operation. But they didn't study the effect of inclusion policy on power consumption of memory. None of the inclusivity models can handle read/write asymmetry effectively. Also dynamically switching inclusivity, does not leverage power minimization opportunities efficiently. Hence, we need a NVM-specific inclusion policy.

4. Details of the solution proposed

a. Theory

LAP uses a combination of non-inclusive and exclusive policies. During an LLC miss, instead of inserting all duplicate data into the LLC, LAP relies on a non-invalidation policy during hits and a replacement policy to keep loop-blocks in the LLC. Only clean victims without duplicates are inserted during L2 evictions, reducing the number of writes to the LLC.

The replacement policy uses set-dueling where a few sample sets may use any baseline policy like LRU and others use loop-block aware policy. In the loop-block aware policy, we replace a LRU non-loop block first followed by loop blocks. Hence, a loop-block gets a higher priority to stay in the LLC.

To identify a loop-block, a loop-bit is used in both L2 and L3. It is reset to zero when data is written or is inserted from main memory. On eviction from L2, if a copy is present in LLC, the clean data would be discarded and the loop-bit stored would be updated. The dirty data and its loop-bit would both be updated. If there is no duplicate copy in the LLC, the clean block and its loop-bit would be inserted into the LLC. When there is an LLC hit, the block would not be invalidated. The loop-bit of the inserted L2 block on the LLC hit would be set to one. Our replacement policy tries to maximize loop-bit 1 blocks in LLC.

To extend LAP to hybrid caches, Lhybrid technique is proposed which tries to store maximum loop-blocks in STTRAM portion and frequently-updating non-loop blocks in SRAM portion. For this, if the victim from L2 is dirty and it hits a block in STT-RAM, STTRAM copy is invalidated and the dirty block is inserted in SRAM. If the victim from L2 does not have a duplicate copy in the LLC, it would also be inserted into the SRAM portion of the LLC. When inserting a block to SRAM, the replacement policy would first select an invalid block from SRAM. If there is no invalid block in SRAM, one block in SRAM would be migrated to STT-RAM or evicted from the LLC. If the incoming block is a loop-block or there are loop-blocks in SRAM, the MRU loop-block would be migrated to STT-RAM. If there is an invalid entry in STT-RAM, the migrated block would replace the invalid block. Otherwise, an LRU nonloop-block would be evicted from STT-RAM, or an LRU loop-block would be evicted if there is no non-loop-block in STT-RAM. On the other hand, if there are no loop-blocks in SRAM, including the incoming block, the LRU block in SRAM would be selected as the replacement victim.

b. Implementation:

i. Simulator and configuration used:

Inclusion policies are modeled by modifying gem5. NVM energy and access latency are modeled by NVSim.
Comparison is done between Non-inclusive, Exclusive policy, FLEXclusion, Dswitch Dynamically, LAP-LRU, LAP-Loop, LAP and Lhybrid.
50 SPEC CPU2006 benchmarks are used.

ii. Metrics of comparison:

Metrics used are energy efficiency, MPKI for performance and coherence traffic.

5. Critique

a. Novelty of the solution

It takes into account read/write energy asymmetry of NVMs. Also, it is extendable to hybrid caches and compatible with data-driven bit-level write reducing schemes for NVMs.

b. Strengths and weakness of the proposed solution

i. Why and where this solution works?

This solution works where the ratio of write energy to read energy of memory is high.

ii. Where it does not work?

c. Reasons on outliers

d. Any possible improvements

6. Regarding related works

a. How to decide whether to quantitatively or qualitatively compare?

b. Most influential or most recent related works:

Paper: <u>DASCA : Dead Write Prediction Assisted STT-RAM Cache Architecture</u>
Authors: Junwhan Ahn et. al.
Venue: HPCA 2014

1. Problem addressed in the paper

To reduce writes at STT-RAM based LLC. why? Only reason is to reduce energy consumption and time it takes to do write operations.

2. Motivation for the problem/how relevant or critical is the problem addressed?
STT-RAM's read and write operations are asymmetric in nature. Write takes longer time than read and same goes for energy consumption. Existing solutions either avoid writing data at STT-RAM cache or reduce write energy per instruction by serving writes from the SRAM buffer, hence they are limited by their size and the buffer also consumes static energy. Some approaches use multi level retention to reduce write energy consumed per write operation at the cost of refreshing STT-RAM cells in intervals. Problem with such approaches is that refresh for blocks will require until they are evicted from cache.

3. Motivation for the solution/insights that helped to develop the solution?

Key observation is the presence of dead writes at LLC can be exploited to reduce the number of writes. Most writes are because of writebacks from L1 (2 level cache hierarchy) to LLC and fills from main memory to LLC. And most of these writes are dead writes, which are never used again.

4. Details of the solution proposed
Bypassing dead writes is the proposed solution.
Contributions are:

1.Concept of Dead Writes and their classification
2.Dead Block Prediction in the context of STT-RAM
3.LLC design to bypass Dead Writes for energy reduction

Proposed Micro-architecture:

(A)Sampler: sample few sets from LLC. Entries are Partial PC, Last Access Type, Partial Tag, Replacement bits, Valid.
(B)Prediction Table: Dead block prediction counter indexed by Last Access type

a. Theory
Dead Write Classification:

(A) Dead-on-Arrival Fills: read miss causes fill. If they are never accessed again then it is dead-on-arrival. Can be bypassed.
(B) Dead-Value Fills: writeback from ULC, as soon as read miss fills the cache. Hence writeback updates this block at LLC become dead-value fills. Do not fill as they will anyway written back.
(C) Closing Writes: Last access is a write back to the block before eviction. Do write to LLC instead bypass writes to main memory.

Keep track of last-touch instructions address using partial PC,
i. Are there any heuristics?
b. Implementation:
i. Which simulator, which data set, state-of-the-art works for comparison, what
is the baseline.
X86 cycle accurate simulator based on Pin.
Comparison with SRAM baseline, STT-RAM baseline, STT-RAM + Sampling DBP, RHCA, OAP.
Workloads: SPEC CPU 2006 only write intensive workloads (sorted in decreasing order of Writes Per Kilo Instruction)
ii. What are the metrics of comparison? How to they measure them.
Energy Consumption:

<u>Single core</u>
DASCA reduces 68% energy consumption relative to STT-RAM baseline.
DASCA reduces 47% energy consumption relative to SRAM baseline
SRAM baseline consumes 63% energy consumption relative to STT-RAM baseline
<u>Multicore</u>
Reduces energy consumption by 62%.

SpeedUp:

<u>Single core</u>
6% performance improvement. 3% reduces memory accesses.

Multicore

14% performance improvement.

Coverage and Accuracy of Predictor table

91% prediction accuracy with 1.4% (average) false positive rate.

5. Your own critique (Hint: Think as if you are the official paper reviewer)
a. Comment on novelty of the solution
Classification of dead writes used to tell whether specific memory requests are dead or alive.
b. Your insight into the strengths and weakness of the proposed solution
i. Why, where this solution works.
ii. Where it does not work
c. Reasons on outliers (related to 5 b ii)
d. Any possible improvements
6. Regarding related works:
a. Think as the author and understand how the author would have picked the related works
i. How to decide whether to quantitatively or qualitatively compare?
b. Make note of the most influential related works, most recent impactful related works.

Paper: Density Tradeoffs of Non-Volatile Memory as a Replacement for SRAM based Last Level Cache
Authors: Kunal Kargaonkar, Ishwar Bhati, Huichu Liu, Jayesh Gaur, Sasikanth Manipatruni, Sreenivas Subramoney, Tanay Karnik, Steven Swanson, Ian Young and Hong Wang
Venue: 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture

1. Problem addressed in the paper

   The paper tries to minimize the write latency of NVM-based LLCs while leveraging high density of NVMs.

2. Motivation for the problem

   SRAM based LLCs are fast but have a lower density than NVMs and hence have a limited capacity. These cannot capture the entire working set of applications these days. Hence, NVMs like STTRAM can be used to replace SRAM for LLCs to increase their capacity. But NVMs have high write energy and high write latency which can lead to performance degradation. Also, long write latency puts pressure on request queues which may stall the pipeline.

3. Motivation for the solution

   To reduce the performance degradation due to high write latency, we need to reduce the number of writes in LLC. For this various techniques have been used, some which stall the writes and some which completely bypass. Other techniques use hybrid cache where most of the writes are absorbed by SRAM. Since, performance is very sensitive to the aggressiveness of bypass, a technique is required which incorporates this.

4. Details of the solution proposed

   a. Theory

This paper proposed two techniques for write minimization - Write Congestion Aware Bypass and Virtual Hybrid Cache.

i) WCAB is used to detect and bypass deadblocks directly to the main memory and hence reducing the number of writes taking place in the LLC. The aggressiveness of the bypass is decided based on the degree of LLC congestion. LLC congestion is obtained by using the running average of the number of pending writes in the request queue for last K intervals (int_write_occ).

Bypassing blocks can lead to reduced hit rates and hence performance degradation. So we need to predict accurately which blocks are dead and can be bypassed. For this, a table with 4 entries corresponding to 4 buckets 0-20%, 21-50%, 51-70% and 71-100% is maintained in L2. Each bucket has a 10b counter. The blocks are mapped to the buckets based on their PC access values. Few observer sets are maintained in L2. Whenever a block is evicted from one of these sets, the counter corresponding to that block is decremented and when a block is recalled from LLC, the counter is incremented.

Each block in the non-observer set is given a 2-bit live score corresponding to the bucket mapped to the PC with positive value. Whenever LLC congestion is above a threshold, write bypassing is turned on. The aggressiveness of bypass is determined byp_score_th obtained from int_write_occ. Any block with live score less the byp_score_th is written directly to main memory.

ii) Virtual Hybrid Cache reduces frequent fills from L2 to LLC by using part of SRAM based L2 to smartly retain such blocks. Exclusive cache have frequent fills upto 50% of total writes in LLC and hence minimizing these is a great advantage.

Write merges in L2 - Frequent dirty trips in LLC can be reduced by keeping frequent dirty blocks in L2 for longer time to allow writes to merge without filling the LLC. A count of how many frequent dirty lines are in each L2 set is kept, and if the count is equal or higher than W, then all lines, including frequent dirty lines, participate in victim selection. Otherwise, dirty lines do not participate in victim selection.

Relaxed Exclusivity - In exclusive caches, clean fills also contribute to frequent fills. To resolve this LAP can be used but it leads to reduced capacity. So, instead, lines in LLC closer to LRU position are duplicated to prevent degradation of the replacement policy. The policy does not update the LRU on duplication, unlike LAP policy which moves duplicated lines to the MRU position. Cache lines hit by store request in LLC are not duplicated. When a read request brings a line into L2 and subsequent store makes it dirty in L1, a hint is sent with the coherence packet to deallocate its copy in LLC to prevent unnecessary duplication.

b. Implementation:

i. Simulator and configuration used:

An in-house modified version of Multi2Sim simulator is used. 20 workloads from SPEC CPU 2006 and HPCG benchmark suites are used for simulation.

The technique is compared to Hybrid LLC with 2MB SRAM and 4MB STTRAM, Hybrid LLC with 1MB SRAM and 6MB STTRAM and LAP.

ii. Metrics of comparison:

The metrics for comparison include percentage reduction in writes, performance and miss rate.

5. Critique

a. Novelty of the solution

The technique takes into account the compromise of hit rate with bypassing and improves performance of the cache.

b. Strengths and weakness of the proposed solution

i. Why and where this solution works?

ii. Where it does not work?

c. Reasons on outliers

d. Any possible improvements

6. Regarding related works

a. How to decide whether to quantitatively or qualitatively compare?

b. Most influential or most recent related works:

Paper: Improving the Lifetime of Non-Volatile Cache by Write Restriction
Author : Sukarn Agrawal, Hemangee K. Kapoor
Venue: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture

1. Problem addressed in the paper

The paper tries to improve the lifetime of NVM based caches by distributing writes among all the ways equally.

2. Motivation for the problem

Since NVMs have higher density, they can serve as a better alternative to SRAM based LLCs. But their limited write endurance restricts their use in LLC. Also, the replacement policies further worsen the situation by biasing the writes towards a few blocks only. Hence, we need a technique that can distribute the writes evenly so that lifetime of the nvm can be increased.

3. Motivation for the solution

Earlier techniques distribute data by migrating heavily written blocks. This was an overhead and degraded performance. So, we need a light technique to handle the same.

4. Details of the solution proposed

a. Theory

Three techniques are proposed - Static Window Write Restriction, Dynamic Window Write Restriction and Dynamic Way Aware Write Restriction.

i) SWWR - The cache is divided into windows with m ways each. For every interval, writing to one of the windows is restricted. The window is selected in a round-robin manner. This way, the writes in each set gets distributed equally.

ii) DWWR - Here, each window is associated with a counter. This counter is incremented whenever there is a write in that window. For every interval, the window with maximum counter value is restricted. The counter of this restricted window is then reset at the end of the interval.

ii) DWAWR - Here, each way is associated with a counter. The counter of a way is incremented on a write to that way. In every interval, n ways with highest counter values are restricted.

b. Implementation:

i. Simulator and configuration used:

Full system simulator GEM-5 is used for evaluation.

Comparison is made between baseline STTRAM, PoLF, WAD and equal chance techniques.
5 benchmarks from PARSEC and 20 benchmarks from SPEC CPU 2006 are used.

ii. Metrics of comparison:

Metrics used for comparison are intra-set write variation coefficient, inter-set write variation coefficient and relative lifetime improvements.

5. Critique
    a. Novelty of the solution
        It is a simple technique for wear leveling with almost nil performance degradation and very little hardware overhead yet it is very effective.
    b. Strengths and weakness of the proposed solution
        i. Why and where this solution works?
            This solution improves the lifetime of every nvm but is not sufficient.
        ii. Where it does not work?
            This solution only deals with intra-set write variation. It cannot reduce inter-set write variation.
    c. Reasons on outliers
    d. Any possible improvements
    6. Regarding related works
    a. How to decide whether to quantitatively or qualitatively compare?
    b. Most influential or most recent related works:
