"ENHANCING THE LIFETIME OF NON-VOLATILE MEMORY BY HYBRID CACHE-REPLACEMENT-POLICY"

A thesis (Phase-I) submitted in partial fulfillment of the requirements for the award of the degree of

Master of Technology

in

Computer Science and Engineering

By

SAMPATH KUMAR S

(206116021)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

TIRUCHIRAPPALLI - 620015

DECEMBER 2017

BONAFIDE CERTIFICATE

This is to certify that the project titled **"ENHANCING THE LIFETIME OF NON-VOLATILE MEMORY BY HYBRID CACHE-REPLACEMENT-POLICY"** is a bonafide record of the work done by

SAMPATH KUMAR S (206116021)

in partial fulfillment of the requirements for the award of the degree of **Master of Technology** in **Computer Science and Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the academic year 2017-2018.

Ms. B. SHAMEEDHA BEGUM
Project Guide

Dr. R. LEELA VELUSAMY
Head of the Department

Project Viva-voce held on ______________________________

Internal Examiner

External Examiner

ABSTRACT

SRAM imposes several challenges on cache design for modern processors because of its large leakage power and low cell density. Non Volatile Memory cache (NVM cache) is a promising alternative to SRAM because of its low leakage power, high density, and non-volatility. However, high write energy and high error rates are reported for NVM caches. This is mainly because of permanent/transient errors due to thermal fluctuations, read disturbance, and process variations.

Spin-Transfer-Torque RAMs (STT-RAMs) are the most promising technology based on Non Volatile Memory for replacing Static RAMs (SRAMs) in on-chip caches. The lifetime of the NVM caches is insufficient because NVM cache cells wear out after the limited number of writes in comparison with SRAM. The lifetime can be extended by reducing the error rate due to stochastic switching in write operations and by reducing the number of writes. Cache replacement algorithms have a major role in the number of write operations into the caches. Due to this fact, it is necessary to redesign cache replacement algorithms for improving the lifetime of STT-RAM caches.

Considering NVMs having low write endurance and the existing cache management policies are having write variation (WV) among the blocks, new cache replacement algorithm has been proposed for reducing the write variations among the block, for reducing the write error rates and for reducing number of writes into the cell.

The main idea is to place the incoming block in a line that incurs the minimum error rate and minimum number of writes in Write operation. This is done by comparing the contents of the incoming block with lines in a cache set. Proposed algorithm is implemented using gem5 Simulator and evaluated with PARSEC benchmark suite. The Proposed algorithm is compared with EqualWrites algorithm. Proposed algorithm improves the lifetime of STT-RAM caches by 27% with about 1% performance overhead.

ACKNOWLEDGEMENTS

First of all, I thank God Almighty for giving me all the blessings to this venture and for helping me in finishing the internship successfully.

This project could not have been successfully completed without the help and cooperation received from some distinguished people.

I express my sincere thanks and gratitude to **Dr. N. Ramasubramanian,** Associate Professor and **Ms. B. Shameedha Begum,** Assistant Professor who guided me with valuable inputs from time to time during my project work and enabled me complete my work successfully.

I am grateful to **Dr. R. Leela Velusamy,** Associate Professor and Head of the Department, Department of Computer Science and Engineering for the facilities she has provided in the department without which the internship could not have been completed.

My hearty thanks to the Department Project Evaluation Committee (DPEC) Chairperson **Dr. B. Nithya** and members of the committee **Dr. E. Sivasankar** and **Ms. S. Jaya Nirmala** for their valuable suggestions during project reviews.

I am also thankful to the faculty and staff members of Department of Computer Science and Engineering, my friends for their support and help. This project work has been carried out by using the facilities of **RISE** lab.

SAMPATH KUMAR S
(206116021)

Contents

List of Figures

Chapter 1

INTRODUCTION

For almost 30 years the memory hierarchy used in computer design has been essentially the same: volatile, high speed memory technologies like SRAM and DRAM used for cache and main memory; magnetic disks for high-end data storage; and persistent, low speed flash memory for storage with low capacity/low energy consumption requirements, such as embedded/mobile devices. Technology scaling limitations of SRAM and DRAM, i.e., large leakage power and low cell density, impose severe challenges on cache design for modern processors. Recent studies imply that Non-Volatile Memory (NVM) technologies such as Phase Change Memories (PCMs), Resistive RAMs (ReRAMs) and Spin-Transfer Torque Random Access Memory (STT-RAM) is a promising alternative to conventional memory technologies for caches. Due to low leakage power, high density, and non-volatility, STT-RAM caches are anticipated to be commercialized in the near future.

1.1 Current landscape of memory systems

An ideal memory system would be fast, cheap, persistent and big (highly dense). Until now, all known memory/storage technologies address only some of these characteristics. Static RAM (SRAM) is very fast, but it's expensive, has low density and is not persistent. Dynamic RAM (DRAM) has better densities and is cheaper (but still expensive) at the cost of being a little slower, and it's not persistent as well. Disks are cheap, highly dense and persistent, but very slow. Flash memory is between DRAM and disk; it is a persistent solid-state memory with higher densities than DRAM, but its write latency is much higher than the later.

Fortunately, it is possible to design a memory system that incorporates all these different technologies in a single memory hierarchy. Such hierarchy allows the creation of a system that approaches the performance of the fastest component, the cost of the cheapest component and energy consumption of the most power-efficient component. This is possible due to a phenomenon known as locality of reference, so named because memory references tend to be localized in time and space.

- If you use something once, you are likely to use it again (temporal locality).
- If you use something once, you are likely to use its neighbor (spatial locality)

This phenomenon can be exploited by creating different memory levels; the first levels are smaller and more expensive, but have fastest access, and the other layers are progressively bigger and cheaper, but have slower access times. Appropriate heuristics are then applied to decide which data will be accessed more often and place them at the first levels, and move the data down on the hierarchy as it ceases to be used frequently.

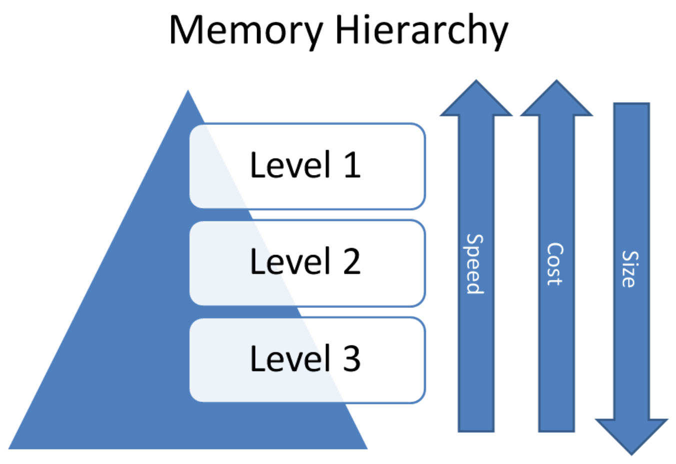


Figure 1.1: Memory Hierarchy

1.2 Non-Volatile Caches

As the number of cores increase, the need for on-chip memory grows in order to reduce the memory access time and delay, and the so-called memory wall issue. However, using SRAM cells as on-chip memory in order to tackle this issue results in significant leakage power and unmanageable temperature increases. To address this issue, non-volatile memory technologies, such as Phase Change Memories (PCMs), Resistive RAMs (ReRAMs) and Spin-Transfer Torque Random Access Memory (STT-RAM) for on-chip memories has gained serious attention. Non-volatile memories (NVMs) like STT-RAMs are emerging memory technologies that have many desirable characteristics such as higher density, near zero leakage power, and high resilience against soft errors.

Despite its advantages, the major challenge of NVMs is their limited write endurance value. Write endurance is defined as the number of times a memory cell can be overwritten, and emerging non-volatile memories commonly have a limited write endurance. The write endurance of both SRAM and DRAM is above 10^{15}, whereas PCM is able to sustain for only 10^{8} writes. ReRAM, another NVM technology having improved write endurance value but it is still around 10^{11}. For STT-RAM, although a prediction of up to 10^{15} write cycles is often cited, the best endurance test result for STT-RAM devices so far less than $4 * 10^{12}$ write cycles.

1.3 STT-RAM

A STT-RAM cell consists of a storage element and a NMOS transistor. The storage element is a Magnetic Tunnel Junction (MTJ) consisting of two ferromagnetic layers separated by a tunnel barrier layer. The magnetic direction of one ferromagnetic layer, called reference layer, is fixed and the direction of the other layer, called free layer, can switch between two states, i.e., parallel and opposite to the reference layer.

The MTJ resistance is low in parallel direction of ferromagnetic layers representing 0 and is high in opposite direction representing 1. The NMOS transistor is a controlling switch to read from and write to MTJ. A low current is applied to MTJ to sense its state on a read operation and a high current with a long duration is applied on a write operation to change the MTJ state.

The main advantages of STT-RAMs over SRAMs are negligible leakage power, higher density, higher immunity to radiation-induced particles strike, and higher security. However, the

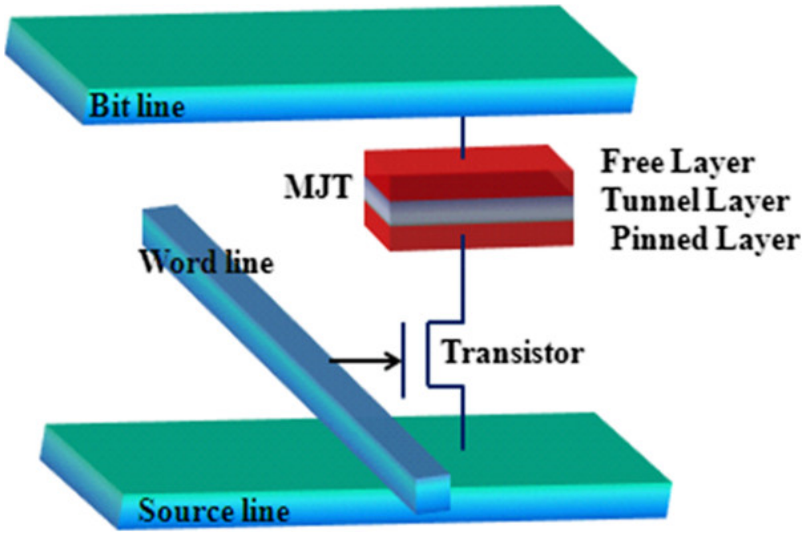


Figure 1.2: Structure of STT-RAM cell

challenges of STT-RAMs need to be addressed to make it applicable in on-chip caches. One of the main challenges in STT-RAMs is their high error rate in write operations. The stochastic switching characteristic of the storing elements, i.e., Magnetic Tunnel Junctions (MTJs), results in high error rate in write operations. This error rate can be reduced by increasing the amplitude and the duration of the write pulses. However, high energy and latency imposed by this approach is unaffordable in the caches. Reducing the error rate of STT-RAM caches with acceptable overheads is a must to replace SRAMs with STT-RAMs.

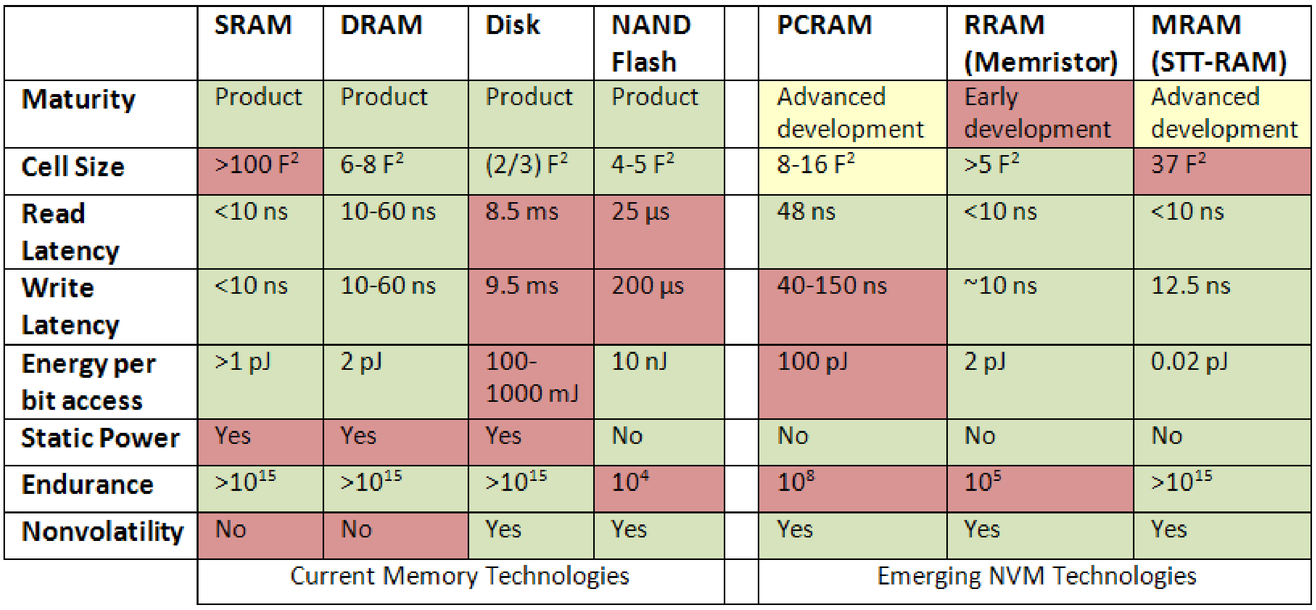
Due to physical characteristics of MTJs, the error rate in STT-RAM cells is asymmetric. For a predetermined amplitude and duration of the write pulse in STT-RAM caches, the switching probability of 0 to 1 transition is different from that in 1 to 0 transition. Therefore, the probability of correct write operation for changing the MTJ state from 1 to 0 is different from changing the MTJ state from 0 to 1. As reported, the write error rate in 0 to 1 transition is higher than that in 1 to 0 transition by about two orders of magnitude.

The write error rate in write operations has a direct relationship with the number of transitions, i.e., the Hamming distance between the incoming block and current block in the cache. In L2-caches, the write operations are mainly due to cache misses and write-backs from L1-caches. Due to the diversity in the contents of the cache lines, the Hamming distances between the incoming block and various lines in a cache set can be significantly different. Therefore, the content of the target cache line that the incoming block will be written into affects the error rate in the write operation. The target cache line is determined by the cache replacement algorithm. This implies that the cache replacement algorithm can significantly affect the STT-RAM error rate.

1.4 Comparison Between Memory Technologies

Based on the following set of criteria, a comparison is made between following set of memory technologies.

- **Maturity**: Whether the technology is currently used in the market or it is in early or later research stages before being commercially mature.
- **Cell size**



	SRAM	DRAM	Disk	NAND Flash		PCRAM	RRAM (Memristor)	MRAM (STT-RAM)
Maturity	Product	Product	Product	Product		Advanced development	Early development	Advanced development
Cell Size	>100 F^2	6-8 F^2	(2/3) F^2	4-5 F^2		8-16 F^2	>5 F^2	37 F^2
Read Latency	<10 ns	10-60 ns	8.5 ms	25 µs		48 ns	<10 ns	<10 ns
Write Latency	<10 ns	10-60 ns	9.5 ms	200 µs		40-150 ns	~10 ns	12.5 ns
Energy per bit access	>1 pJ	2 pJ	100-1000 mJ	10 nJ		100 pJ	2 pJ	0.02 pJ
Static Power	Yes	Yes	Yes	No		No	No	No
Endurance	$>10^{15}$	$>10^{15}$	$>10^{15}$	10^4		10^8	10^5	$>10^{15}$
Nonvolatility	No	No	Yes	Yes		Yes	Yes	Yes
	Current Memory Technologies					Emerging NVM Technologies		

Figure 1.3: Comparison of memory technologies

- **Write Latency**: The speed for writing values to a memory cell.
- **Read Latency**: The speed for readings values from a memory cell.
- **Endurance**: The number of write cycles that a memory cell endures before eventually wearing out.
- **Energy**: Energy spent per bit access.
- **Static Power**: Whether power needs to be spent while not accessing the memory device.
- **Non-volatility**: Whether the memory technology is volatile or not.

Chapter 2

LITERATURE OVERVIEW

Driven by the trends of increasing core-count and bandwidth-wall problem, the size of last level caches has greatly increased, and hence the researchers have explored non-volatile memories (NVMs) that provide high density and consume low-leakage power. Since NVMs have low write endurance and the existing cache management policies are write variation (WV) unaware, effective wear-leveling techniques (WLTs) are required for achieving reasonable cache lifetimes using NVMs. Many studies addressed the above-mentioned reliability concerns in STT-RAM and tried to either extend the lifetime of the caches or to provide more effective protection against errors. Main approaches for increasing the STT-RAM caches lifetime are to reduce write accesses to them, wear-leveling using hybrid memory cells and data coding.

2.1 Inter and Intra-Set Variations

Write variation is a significant concern in designing any cache/memory subsystems with a limited write endurance. Large write variation can greatly degrade the product lifetime because only a small subset of memory cells that experience the worst-case write traffic can result in an entire dead cache/ memory subsystem even when the majority of cells are far from wear-out. The objective of cache wear-leveling is to reduce write variations and make write traffic uniform. To quantify the cache write variation, we first define the coefficient of inter-set variations (InterV) and the coefficient of intra-set variations (IntraV) as follows,

$$InterV = \frac{1}{W_{aver}} \sqrt{\frac{\sum_{i=1}^{N} \left(\sum_{j=1}^{M} w_{i,j}/M - W_{aver} \right)^2}{N-1}}$$

$$IntraV = \frac{1}{W_{aver}.N} \sum_{i=1}^{N} \sqrt{\frac{\sum_{j=1}^{M} \left(w_{i,j} - \sum_{j=1}^{M} w_{i,j}/M \right)^2}{M-1}}$$

where $w_{i,j}$ is the write count of the cache line located at set i and way j, W aver is the average write count defined as:

$$W_{aver} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} w_{i,j}}{NM}$$

N is the total number of cache sets, and M is the number of cache ways in one set. If every $w_{i,j}$ has the same value, then InterV = IntraV = 0. In short, InterV is defined as the CoV (coefficient of variation) of the average write count within cache sets, and IntraV is defined as the average of the CoV of the write counts cross a cache set.

2.2 Probabilistic Set Line Flush

PoLF (Probabilistic Set Line Flush), a technique used to reduce the intra-set variations in NVMs. PoLF is to flush hot data probabilistically instead of deterministically. It maintains one global counter to count the number of write hits to the entire cache, and it flushes a cache line when the counter saturates no matter whether the cache line to be flushed is hot or not. Although there is no guarantee that the hottest data would be flushed as we desire, the probability of PoLF selecting a hot data line is high: the hotter the data is, the more likely it will be selected when the global counter saturates. Theoretically, PoLF is able to flush the hottest data in a cache set most of the time, and the big advantage of PoLF is that it only requires one global counter.

2.3 Write Smoothing

Write Smoothing is a technique for improving cache lifetime by mitigating intra-set write variation. Write Smoothing logically divides the cache-sets into multiple modules and records number of writes in each way for all the sets. If the intra-set write variation in a module is larger than a threshold, then the most heavily written cache ways can be temporarily made "unavailable" which will shift the write pressure on the remaining ways. Different cache ways are made unavailable in rotation and this leads to wear-leveling which improves the cache lifetime. Write Smoothing does not require static profiling or modification of program binary and its overhead is very small.

2.4 EqualWrites

EqualWrites is a technique for improving cache lifetime by mitigating intra-set write variation. EqualWrites works on the key idea that if the difference between the number of writes to two blocks in a cache set is larger than a threshold, it indicates large intra-set WV and to mitigate it, the data item in these blocks can be swapped so that future writes can be redirected from a hot block to a cold block to achieve wear leveling. This leads to improvement in cache lifetime. EqualWrites does not require offline profiling or modification of program binary or including set-index bits as part of the tag or any floating-point hardware for computation of coefficient of WV.

2.5 AYUSH

The features and limitations of both SRAM and NVM (non-volatile memory) technologies have led the researchers to study SRAM-NVM way-based hybrid last level caches (LLCs). Since large leakage power consumption of SRAM allows including only few SRAM ways, the small write-endurance of NVM may still lead to small lifetime of these hybrid caches. AYUSH is a technique proposed for improving the lifetime of SRAM-NVM hybrid caches.

These caches use a few SRAM ways along with large number of NVM ways to leverage the high endurance of SRAM along with high capacity (density) and low leakage of NVM. AYUSH migrates data only on write-hits and not on read hits. Thus, by not moving read-intensive blocks into SRAM, AYUSH makes best use of SRAM ways for improving cache lifetime. By proactively migrating write-intensive data to SRAM ways, AYUSH adapts to changing working set much faster than LRU policy. NVM cache lifetime enhancement techniques use either in-cache data-movement or data-invalidation.AYUSH uses in-cache data movement, which does not increase writes to main memory and thus, it does not harm performance or energy efficiency.

2.6 Data Comparison Write Scheme

Data Comparison Write (DCW) Scheme is proposed to reduce the large write power. It performs the read operation before the write operation to know the previously stored data in the selected PRAM cell. If the input data and the previously stored data are the same, no write operation performs. If not, the write operation is the same as the conventional write scheme. When the probabilities of four cell data transitions are 1/4, the DCW scheme does not consume the write power for two cases ($0 \to 0, 1 \to 1$). Therefore, the average power of the DCW scheme is half of that of the conventional write scheme.

2.7 Flip-N-Write

Flip-N-Write technique is to replace a PRAM write operation with a more efficient read-modify-write operation. On a write, a quick bit-by-bit inspection of the original data word and the new data word is done to find the hamming distance. Depends on the Hamming distance, Flip-N-Write writes either the new data word or the “flipped” value of it depends on the hamming distance. Flip-N-Write introduces a single bit associated with each PRAM word to indicate whether the PRAM word has been flipped or not.

2.8 Least Error Rate Replacement Algorithm

One of the major problems in STT-RAM is the high error rate due to stochastic switching during write operations. Cache replacement algorithms have a major role in the number of write operations into the caches. Least-Error-Rate (LER) Replacement algorithm is a modified cache replacement algorithm introduced to reduce the error rate in L2 caches. The main idea is to place the incoming block in a line that incurs the minimum error rate during write operation. This is done by comparing the contents of the incoming block with lines in a cache set.

Chapter 3

PROPOSED SOLUTION

In this paper, new Cache Replacement Policy has been proposed to overcome a major challenge in STT-RAM caches, ie., the high error rate during the Write operation. Prior to introducing the proposed algorithm, we first investigate the importance of the error rate as well as the contribution of transition directions in total write error rate. The switching operation of STT-RAM cells has a stochastic characteristic. This characteristic makes the STT-RAMs write operations error prone. Considering the stochastic switching, the probability of correct write operation is increased with increasing in the amplitude and duration of the write pulse. The required amplitude and duration of the write pulses for switching the MTJ cells from 0 to 1 is significantly higher than switching MTJ from 1 to 0. Thus, for a predetermined amplitude and duration of the write pulse in STT-RAM caches the error rate of STT-RAM write operation is asymmetric by the direction of transitions.

Fig 3.1 shows the distribution of 0 to 1 transitions and 1 to 0 transitions and their contributions in write errors. Based on predefined results, the error rate in MTJ switching from 0 to 1 is about two orders of magnitude higher than MTJ switching from 1 to 0. On the other hand, Fig. 3.1 depicts that the contribution of 0 to 1 and 1 to 0 transitions in total STT-RAM cells switching are almost the same. Significantly higher error rate in one transition direction over the other direction when the numbers of transitions in these directions are the same implies that the majority of write errors is due to 0 to 1 transition. This is illustrated for the L2-cache under the workload of PARSEC benchmark suite. As depicted, the contribution of 0 to 1 transitions in total error rate is 100 larger than the contribution of 1 to 0 transitions.

The location of the incoming block to be placed in the target set will be finalized by cache replacement policy. In order to reduce the error rate during the write operation, it is necessary to redesign the cache replacement policy. Since the error rate of 0 to 1 transition is 100x larger than 1 to 0 transition, it is important to reduce the number of 0 to 1 transition in the proposed algorithm. If the number of 0 to 1 transitions are reduced, the probability of error rates will be less when compared with the conventional algorithm.

The proposed algorithm is broadly classified into two major modules.

- Write Phase
- Read Phase

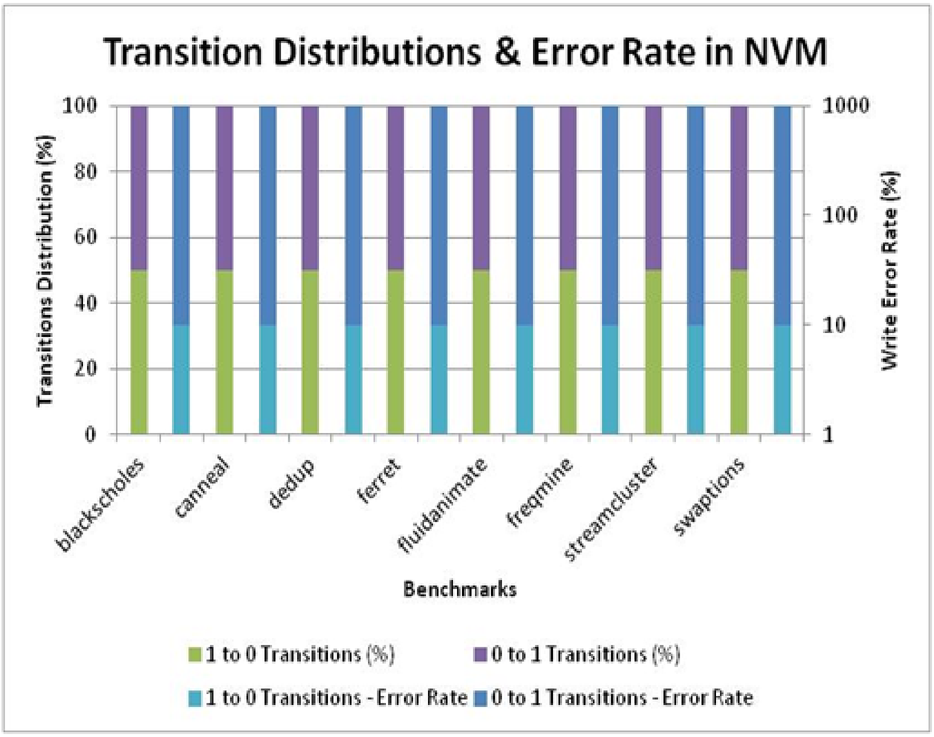


Figure 3.1: Transition Distribution and write error rate in STT-RAM

3.1 Write Phase

The proposed algorithm is based on Wear-leveling technique to reduce the intra-set variations. Wear-leveling Techniques are used for pro-longing the service lifetime of erasable computer storage. This technique ensures writes are distributed across the cache sets and blocks. The proposed algorithm is added as an enhancement on the existing wear-leveling algorithm "Equal-Writes" to reduce the number of writes and to reduce the write error rates. Data comparison write scheme is considered during the write operation. Before writing the new data, read operation will be perform to know the previously stored data. If the input data and previously stored data are same, no write operation performs. If not, the new data will be written to the particular cell.

Algorithm 1 Proposed Cache Replacement Policy (Part 1)

```
1:  begin
2:  avail = 'N'
3:  target_set = extractSet(new_block)
4:  // Checking whether any redirection block available in target set
5:  for ($i$ = $assoc$ - 1 $to$ 0) do
6:      if ($block[i].counter$ == 0) then
7:          avail = 'Y'
8:          break from for loop
9:      end if
10: end for
```

3.1.1 Inverse Flag

Inverse Flag is one bit information stored in every block to indicate whether the data is stored in original format or inverted format. With the help of this bit, at most half the number of 0 to 1 transitions will be guaranteed in the cache line in the worst case.

3.1.2 Counter

Counter is 3 bit information (If the threshold = 8), 4 bit (If the threshold = 16) stored in every block to ensure writes are evenly distributed within the set. It is an intelligent way of distributing the writes within the set, Instead of storing the count of actual number of writes. This requires only 3 bits, but storing actual number of writes in the block will incurs large number of bits and more computation. Counter will ensure the write difference between the block within the set will be maximum of threshold value. Initially, all the block counter will be initialized with threshold/2 (Threshold value = 8,12,16 etc).

3.1.3 Identification of cold block

The proposed algorithm is divided into 3 modules. Algorithm 1 is used to identify the cold block (Block which is having lesser number of writes) in the target set, If the target block is hot block (Block which is having higher number of writes), it can be redirected to the cold block. This can be identified with the help with the help of counter value stored in each block. If the counter value is equal to 0 indicates the number of writes difference between the current block and target block is greater than threshold. avail flag will be used to indicate the availability of the cold block.

- If avail = ‘N’, redirection block is not available within the target set.
- If avail = ‘Y’, redirection block is available within the target set.

3.1.4 Identification of victim block

Algorithm 2 is used to identify the block which incurs less write error rate. As seen in previous sections, number of 0 to 1 transitions is directly proportional to the write error rate. For reducing the write error rate, it is mandatory to reduce the number of 0 to 1 transitions during the write operation.

The victim block will be searched among the Invalid lines, in order to improve the miss rates. If no such lines are available, the victim block will be searched among the valid lines. Number of 0 to 1 transitions required between the incoming data and existing data in the original format and inverted format will be computed. Based on the minimum number of 0 to 1 Hamming distance and inverse 0 to 1 Hamming distance output, inverse flag will be updated.

- If 0 to 1 Hamming distance is minimum, inversion flag will be reset. Incoming data will be stored in actual format.
- If inverted 0 to 1 Hamming distance is minimum, inversion flag will be set. Incoming data will be stored in inverted format.

3.1.5 Reducing Intra-set Variation

Algorithm 3 is used to reduce the Intra-set variation between the blocks within the set. The victim block identified in algorithm 2 will be validated to maintain intra-set variation.

- If the target block counter value is not equal to threshold-1, Increment the counter value and return the target block.
- If the target block counter is equal to threshold-1 and no redirection block is identified in algorithm 1, Decrement the counter value of all the blocks within the set except the target block by one and return the target block.
- If the target block counter is equal to threshold-1 and redirection block is identified in algorithm 1, Identify target block among the redirection block (having the counter value as 0) which is having minimum 0 to 1 hamming distance or inverse 0 to 1 hamming distance is chosen, update the inverse flag accordingly.

Algorithm 2 Proposed Cache Replacement Policy (Part 2)

```
 1: find_target = 0
 2: target = NULL
 3: // Searching among the invalid block in target set
 4: for ($i$ = $assoc$ - 1 $to$ 0) do
 5:    if ( $0to1trans$ $is$ $minimum$ and $block[i]$ == $Invalid$ ) then
 6:       target = block[i]
 7:       inv = ‘N’
 8:       find_target = 1
 9:    else
10:       if ( $Inverse$ $0to1trans$ $is$ $minimum$ and $block[i]$ == $Invalid$ ) then
11:          target = block[i]
12:          inv = ‘Y’
13:          find_target = 1
14:       end if
15:    end if
16: end for
17: // Searching among the valid block in target set
18: if ($find\_target$ == 0) then
19:    for ($i$ = $assoc$ - 1 $to$ 0) do
20:       if ( $0to1trans$ $is$ $minimum$) then
21:          target = block[i]
22:          inv = ‘N’
23:       else
24:          if ( $Inverse$ $0to1trans$ $is$ $minimum$) then
25:             target = block[i]
26:             inv = ‘Y’
27:          end if
28:       end if
29:    end for
30: end if
```

- If the new target block is Invalid, Mark the target block as Invalid, Make the new target block as clean or dirty appropriately, return the new target block.

- If the new target block is valid, Write the data stored in new target block to the target block and mark it as clean or dirty appropriately. return the new target block.
- Make the counter value of target and new target block as threshold/2.

Algorithm 3 will ensure the Intra-set variation between the blocks within the set.

3.2 Read Phase

During the read phase, Inverse flag will be used to identify whether the data is stored in original format or inverted format.

- If the Inverse flag is 1, the data should be returned in reverse format.
- If the Inverse flag is 0, the data can be returned in original format.

Algorithm 3 Proposed Cache Replacement Policy (Part 3)

```
if ($target.counter == threshold - 1$ and $avail ==$ ‘$N$’) then
    for ($i = assoc$ - 1 $to$ 0) do
        if ($block[i] <> target$ ) then
            block[i].counter–
        end if
    end for
else
    if ($target.counter == threshold - 1$ and $avail ==$ ‘$Y$’) then
        new_target = minimum 0to1trans and $target.counter == 0$
        if ($block[new\_target == Valid)$ then
            write Data[new_target] to Data[target] and mark clean or dirty appropriately
            write new data to Data[new_target] and mark dirty
        else
            write new data to Data[new_target] and mark valid and dirty
            mark Data[target] as Invalid
        end if
        block[target].counter = block[new_target].counter = threshold/2
        target = new_target
    else
        block[target].counter++
    end if
end if
return target
end
```

Chapter 4

RESULTS AND ANALYSIS

To evaluate the proposed algorithm, gem5 is used as the simulation platform and PARSEC benchmark suite is used as the workload. The simulations are performed for one billion instructions. The energy consumption information of STT-RAM cache are retrieved from mcpat. The simulations are performed on the detailed model of the ALPHA processor provided by gem5. The configurations used for the simulations,

- CPU Model = ALPHA detailed
- L1 Replacement Algorithm = LRU
- Number of CPUs = 4 (quad-core)
- L1D Cache Size = 32kB
- L1D Associative = 4
- L2 Cache Size = 1MB
- L2 Associative = 16
- L2 Replacement Algorithm = LRU, EqualWrites and Proposed algorithm
- Block Size = 64B

4.1 PARSEC Benchmark Suite

The Princeton Application Repository for Shared-Memory Computers (PARSEC) is a benchmark suite composed of multi-threaded programs. The suite focuses on emerging workloads and was designed to be representative of next-generation shared-memory programs for chip-multiprocessors. The following benchmark suite part of the PARSEC suite are used for the evaluation of the proposed solution:

4.1.1 Blackscholes

This application is an Intel RMS benchmark. It calculates the prices for a portfolio of European options analytically with the BlackScholes partial differential equation(PDE). There is no closed-form expression for the BlackScholes equation and as such it must be computed numerically.

4.1.2 Canneal

This kernel was developed by Princeton University. It uses cache-aware simulated annealing (SA) to minimize the routing cost of a chip design. Canneal uses fine-grained parallelism with a lock-free algorithm and a very aggressive synchronization strategy that is based on data race recovery instead of avoidance.

4.1.3 Dedup

This kernel was developed by Princeton University. It compresses a data stream with a combination of global and local compression that is called 'deduplication'. The kernel uses a pipelined programming model to mimic real-world implementations. The reason for the inclusion of this kernel is that deduplication has become a mainstream method for new-generation backup storage systems.

4.1.4 Ferret

This application is based on the Ferret toolkit which is used for content-based similarity search. It was developed by Princeton University. The reason for the inclusion in the benchmark suite is that it represents emerging next-generation search engines for non-text document data types. In the benchmark, we have configured the Ferret toolkit for image similarity search. Ferret is parallelized using the pipeline model.

4.1.5 Fluidanimate

This Intel RMS application uses an extension of the Smoothed Particle Hydrodynamics (SPH) method to simulate an incompressible fluid for interactive animation purposes [19]. It was included in the PARSEC benchmark suite because of the increasing significance of physics simulations for animations.

4.1.6 Freqmine

This application employs an array-based version of the FP-growth (Frequent Pattern-growth) method for Frequent Itemset Mining (FIMI). It is an Intel RMS benchmark which was originally developed by Concordia University. freqmine was included in the PARSEC benchmark suite because of the increasing use of data mining techniques.

4.1.7 Streamcluster

This RMS kernel was developed by Princeton University and solves the online clustering problem. streamcluster was included in the PARSEC benchmark suite because of the importance of data mining algorithms and the prevalence of problems with streaming characteristics.

4.1.8 Swaptions

The application is an Intel RMS workload which uses the Heath-Jarrow-Morton (HJM) framework to price a portfolio of swaptions. Swaptions employs Monte Carlo (MC) simulation to compute the prices.

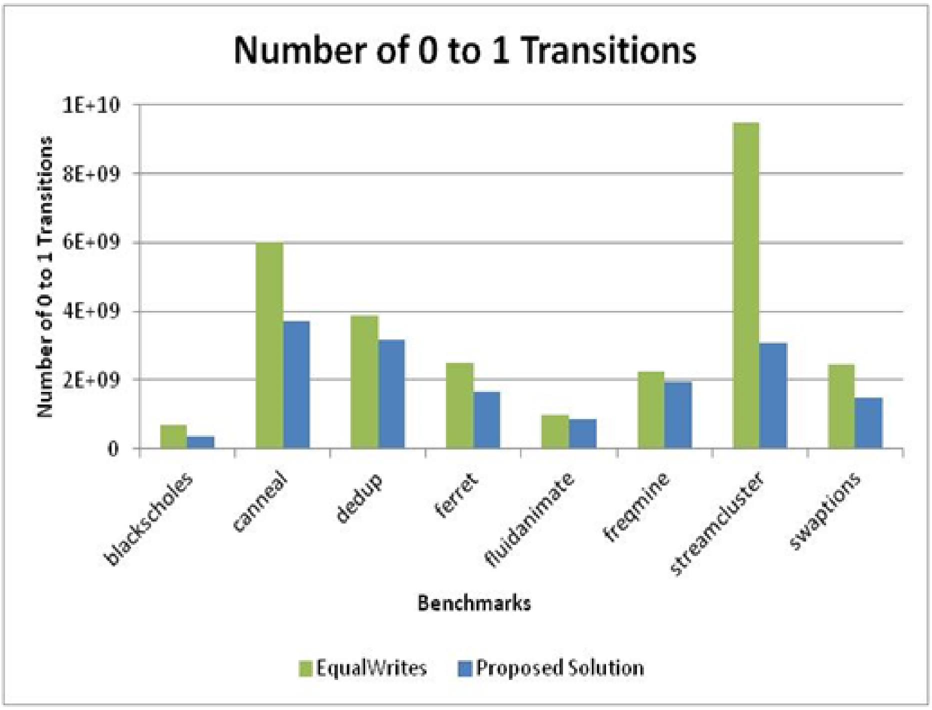


Figure 4.1: Comparison of number of 0 to 1 Transitions

4.2 Evaluation Metrics

Baseline is a cache that uses LRU replacement policy, but does not use any scheme for improving cache lifetime. Results are shown on the following metrics and compared the results with existing wear-leveling technique EqualWrites.

- Number of 0 to 1 Transitions
- Relative Lifetime
- Distribution of Writes
- Relative Performance

4.2.1 Number of 0 to 1 Transitions

In the Proposed algorithm, Early Write Termination (EWT) method is considered for the simulations. The error rate reduction in Proposed solution is proportional to the reduction in the number of 0 to 1 transitions. Fig. 4.1 shows comparison of the total number of 0 to 1 transitions in L2 Cache in EqualWrites and Proposed algorithm. It is noted that, the proposed algorithm reduces the total number of 0 to 1 transitions by about 42.1% when compared with EqualWrites method. In the best case, Streamcluster, a workload related to Data mining algorithms is getting major advantages, reducing a maximum of 67% 0 to 1 transitions when compared with EqualWrites. In the worst case, Freqmine reduces the number of 0 to 1 transitions by 11.7%.

Inorder to get the clarity in the reduction of 0 to 1 Transition, the EqualWrites 0 to 1 Transitions is normalized to 100%, while proposed solution 0 to 1 Transitions is calculated accordingly. Fig 4.2 shows the Normalized 0 to 1 Transitions. The efficiency of Proposed solution in reducing

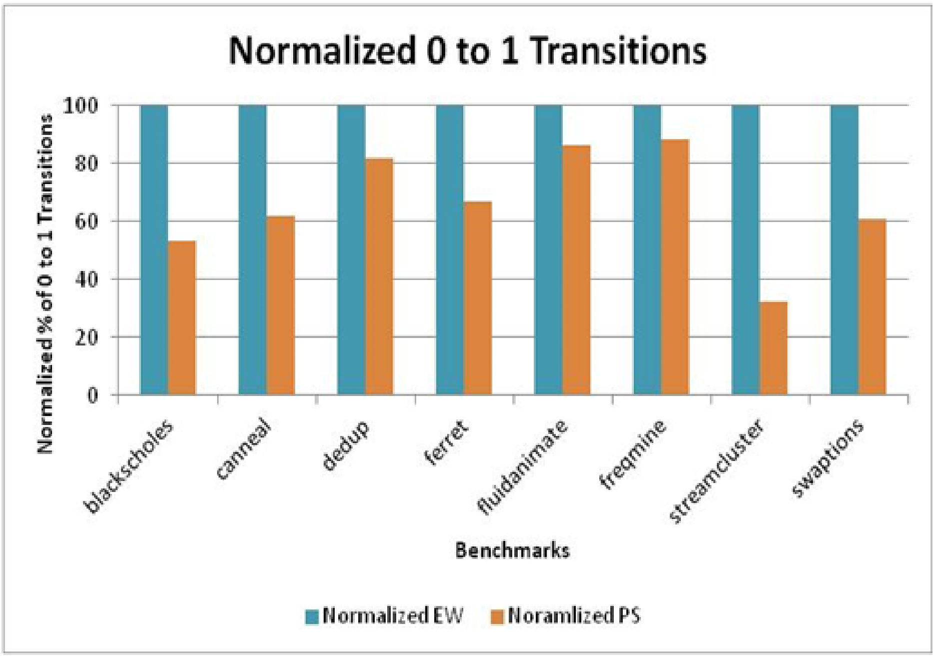


Figure 4.2: Comparison of Normalized number of 0 to 1 Transitions

the error rate depends on the data value patterns of the workloads in the cache lines. For workloads with higher diversity in the contents of the cache lines, the incoming block incurs higher variation in the number of transitions when written into various lines.

Fig 4.3 shows the average number of 0 to 1 Transitions in the cache line. Average number of 0 to 1 transitions are calculated by the ratio of total number of 0 to 1 transitions to the total number of writes to the cache block.

$$Avg\ number\ of\ 0\ to\ 1\ Transitions = \frac{Total\ Number\ of\ 0\ to\ 1\ Transitions}{Total\ Number\ of\ writes}$$

There is a significant decrease in the number of 0 to 1 transitions in our proposed solution. Since number of 0 to 1 transitions is proportional to the write error rate. The decrease in the number of 0 to 1 transtions will reduce the write error rate. Maximum of 36% reduction in Streamcluster, a workload related to Data mining algorithm. In the worst case, Fluidanimate and freqmine reduces 6% and 3% average number of 0 to 1 transitions in cache line in cache line block. On an average, 21.42% reduction in the average number of 0 to 1 transitions in the cache block.

4.2.2 Relative Lifetime

The reduction in the error rate, will have significant impact in the lifetime of the cache. The reduction in the write error rate will improve the lifetime of the cache. The relative lifetime is defined as the inverse of the maximum number of writes on any cache block on using that technique (or baseline).

$$Relative\ Lifetime = \frac{Lifetime_{Baseline}}{Lifetime_{Proposed}}$$

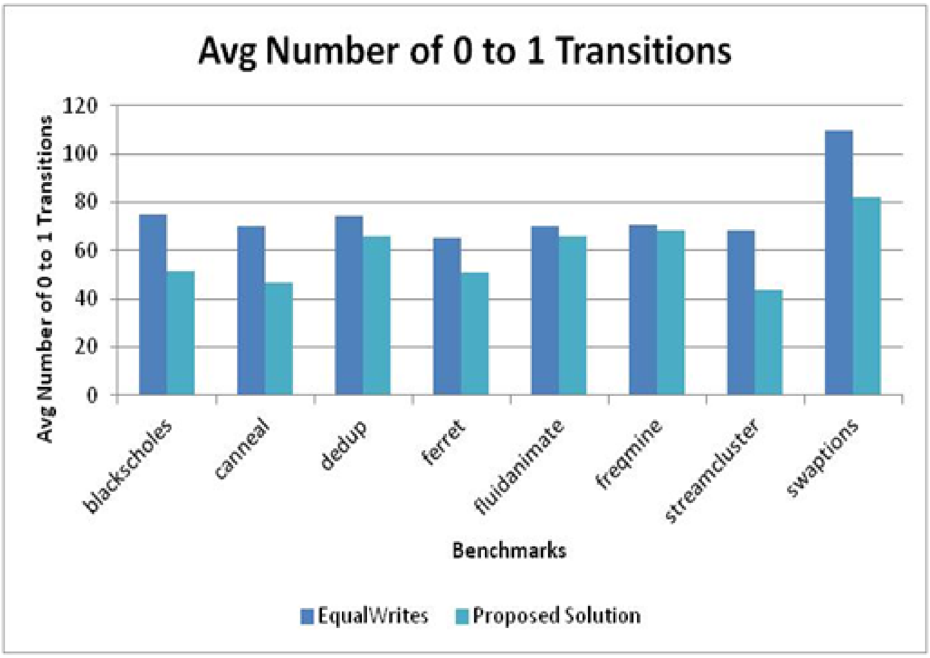


Figure 4.3: Comparison of average number of 0 to 1 Transitions in cache line

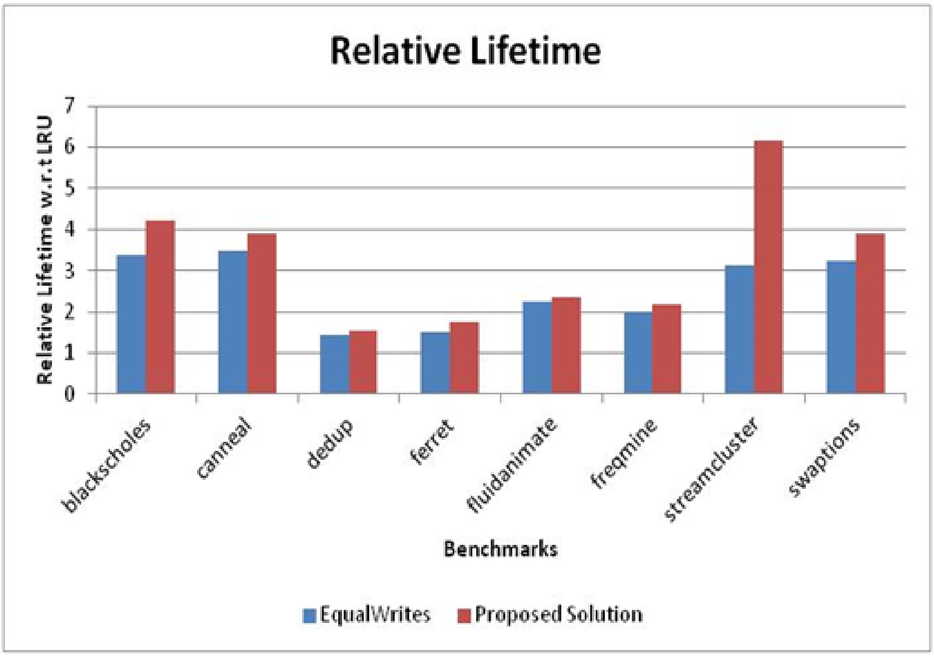


Figure 4.4: Comparison of Relative Lifetime w.r.t LRU

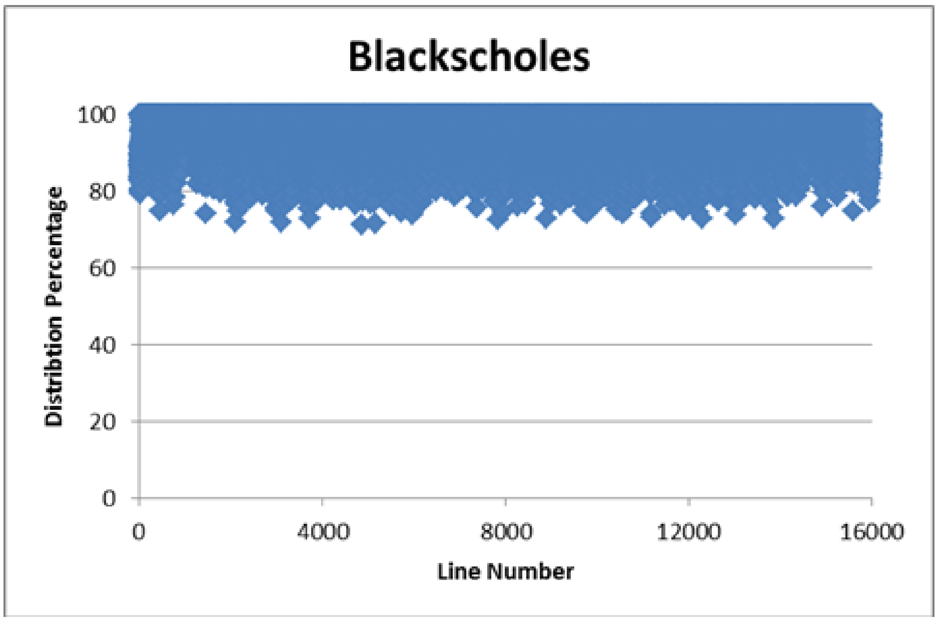


Figure 4.5: Blackscholes - Distribution of Writes

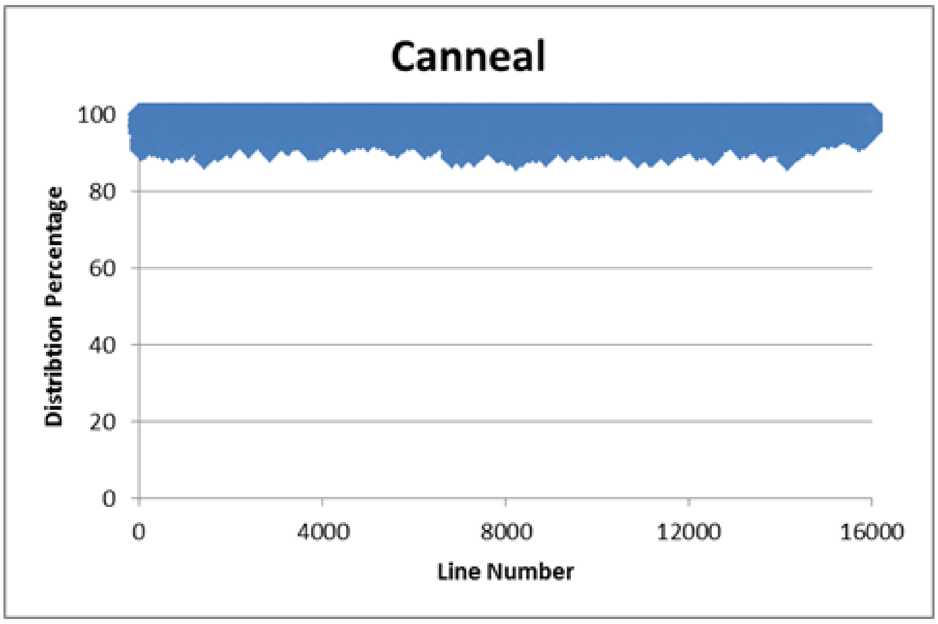


Figure 4.6: Canneal - Distribution of Writes

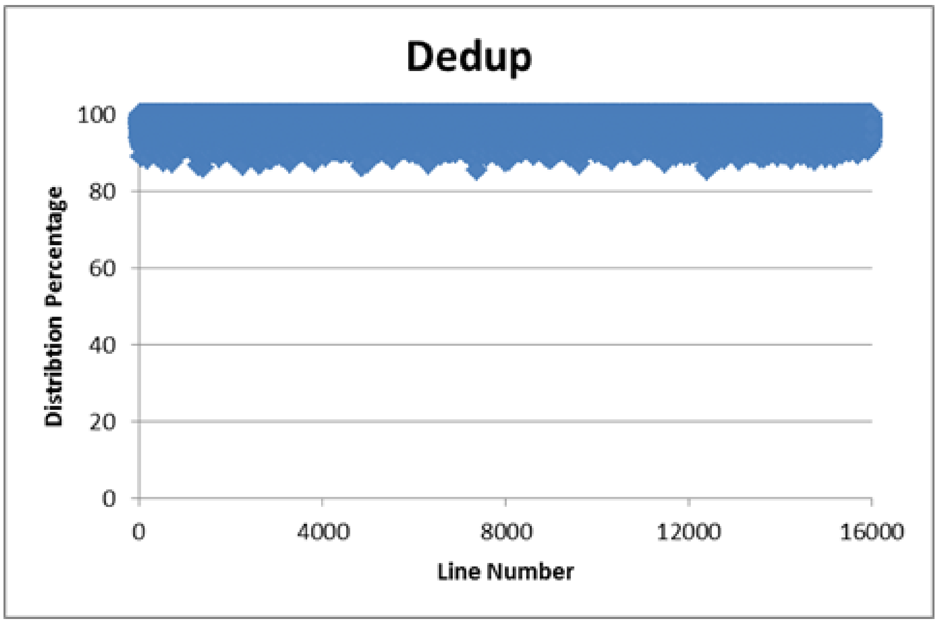


Figure 4.7: Dedup - Distribution of Writes

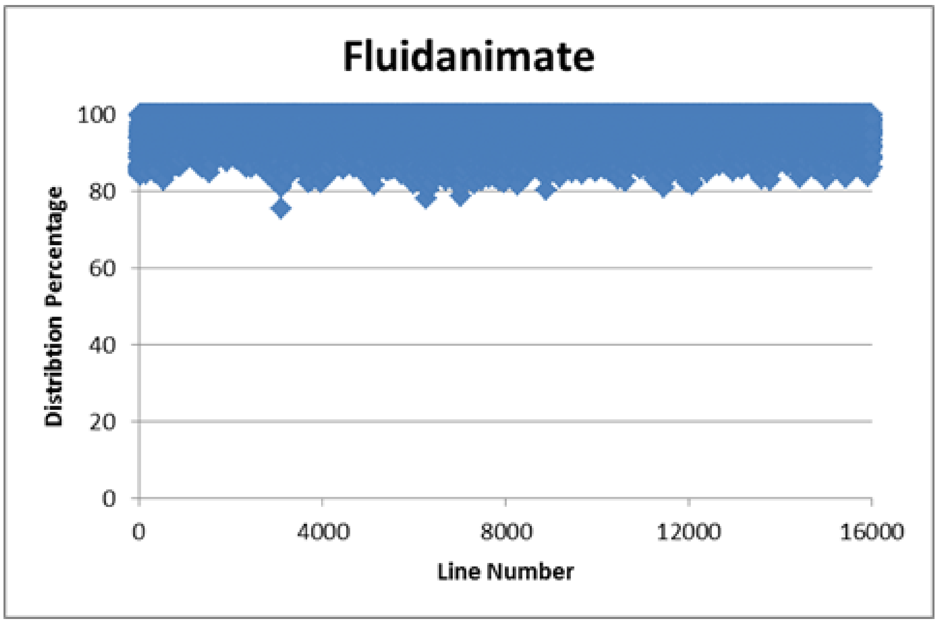


Figure 4.8: Fluidanimate - Distribution of Writes

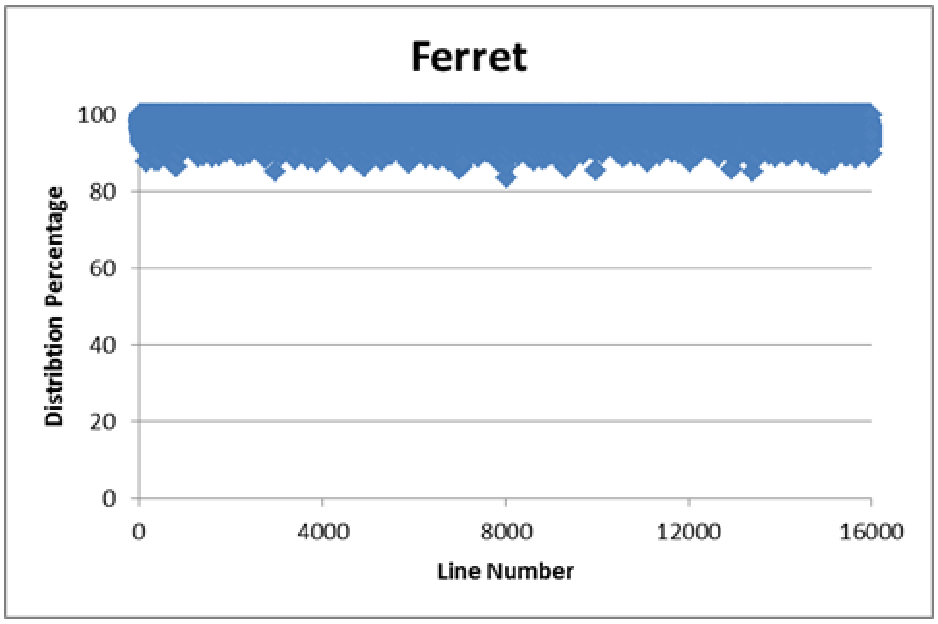


Figure 4.9: Ferret - Distribution of Writes

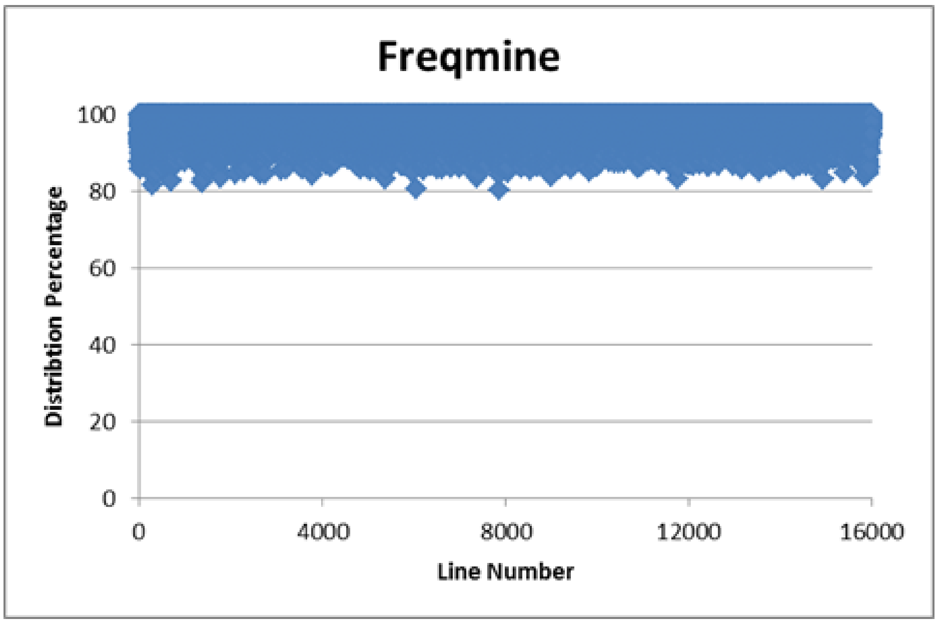


Figure 4.10: Freqmine - Distribution of Writes

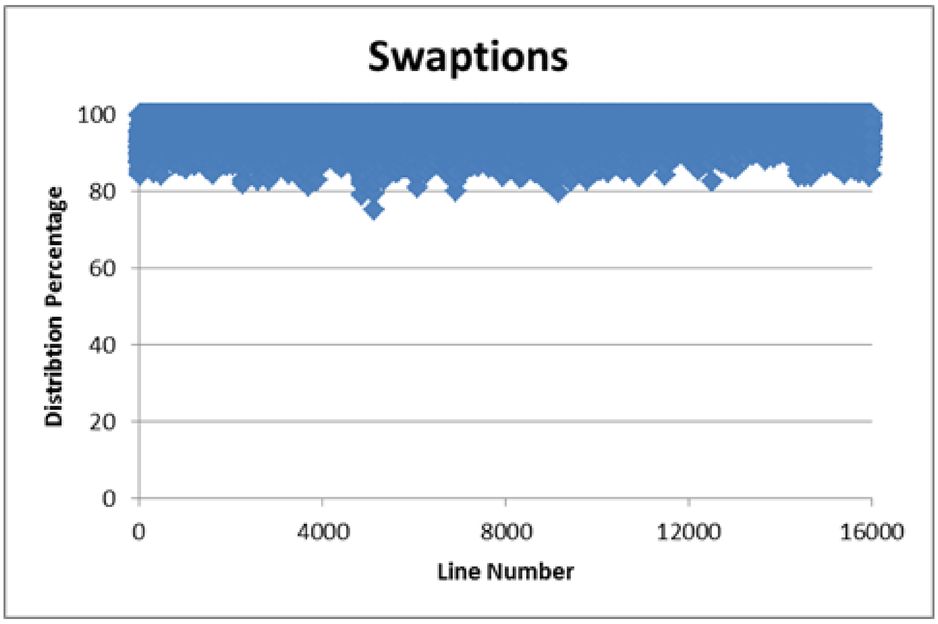


Figure 4.11: Swaptions - Distribution of Writes

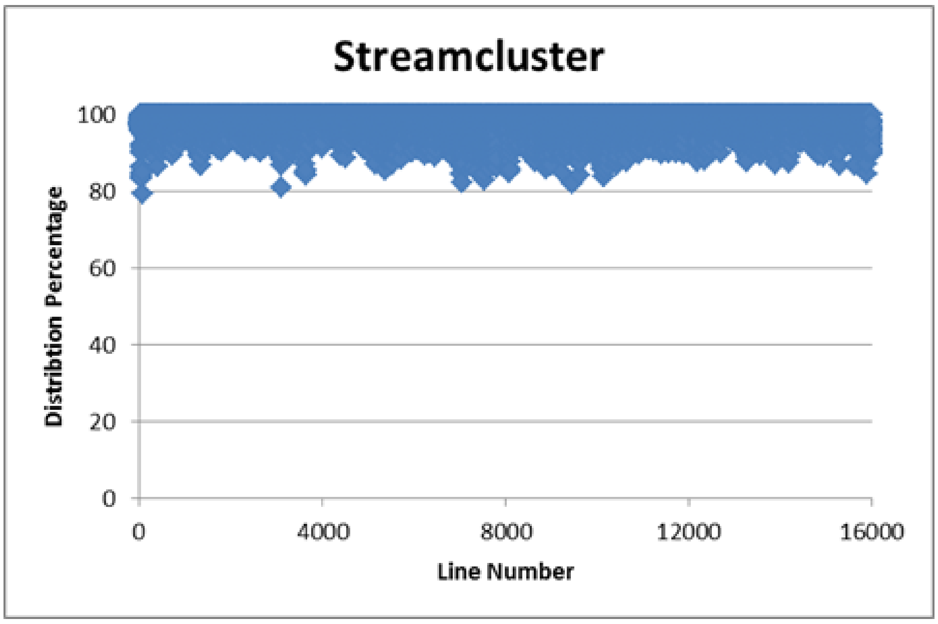


Figure 4.12: Streamcluster - Distribution of Writes

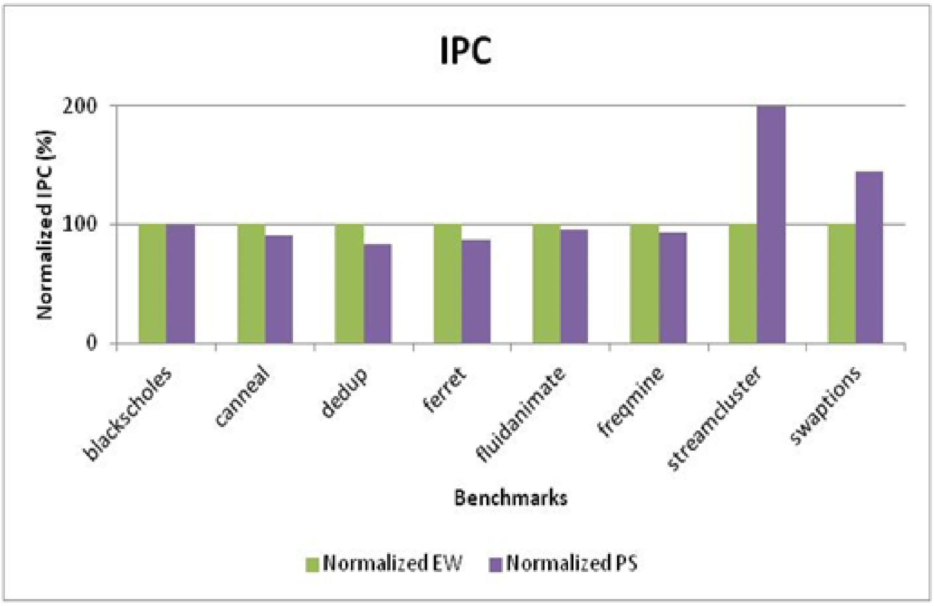


Figure 4.13: Comparison of IPC

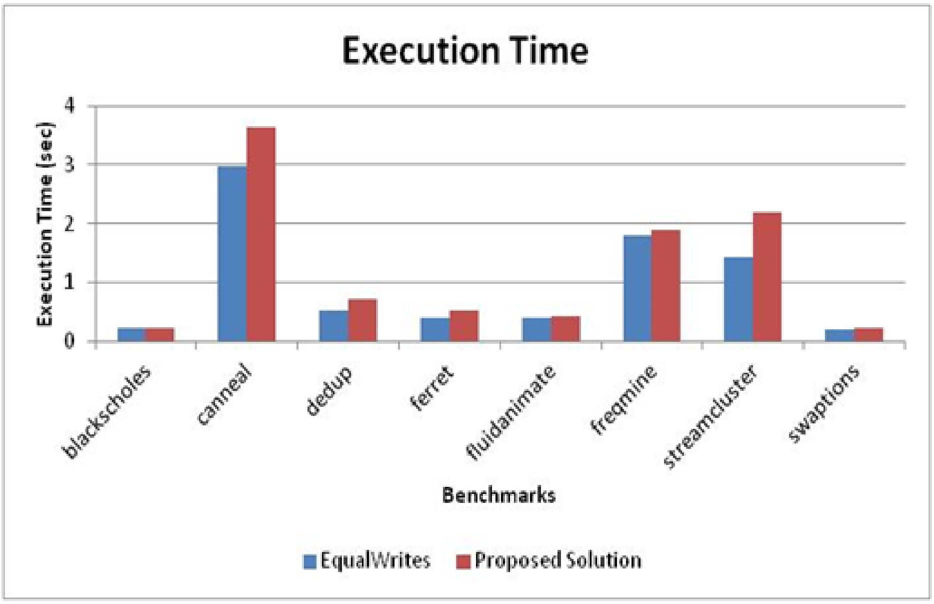


Figure 4.14: Comparison of Execution Time

Baseline is a cache that uses LRU replacement policy, but does not use any scheme for improving cache lifetime. Fig 4.4 shows the relative lifetime of proposed algorithm with respect to LRU. The proposed algorithm is compared with EqualWrites. On an average, 27% improvement in the lifetime of the cache when compared with EqualWrites technique. In the best case, Streamcluster is producing 96% Improvement, while Fluidanimate is producing 6% improvement in the cache lifetime at the worst case. Streamcluster reduces the number of 0 to 1 transitions by about 67%, So it account for the maximum improvement in the lifetime of the cache. while fluidanimate account for the reduces the number of 0 to 1 transitions by about 11%, So it account minimal improvement in the lifetime when compared relatively with respect to the other benchmarks.

4.2.3 Distribution of Writes

Fig 4.5 - 4.12 describes the distribution of writes across the block within the set. The total number of writes in each block is computed for analyzing the distribution of writes among the blocks. It is observed that all the benchmarks having the distribution percentage of more than 90% at the best cas and atleast 80% distribution among the block within the set at the worst case.

4.2.4 Relative Performance

The proposed algorithm provide almost same performance compared with EqualWrites as shown in fig 4.13. For evaluating the performance, IPC (Instruction Per Clock) is considered. Blackscholes is providing almost same IPC in proposed algorithm, when compared with EqualWrites. At the worst case, ferret is producing 14% decrease in the IPC. At the best case, Streamcluster is providing 101% increase in IPC, while swaptions is providing 44% increase in IPC. On an average, 11% increase in performance for the proposed solution, 1% decrease in performance if streamcluster is excluded. This is due to the additional computation incurred during the identification of the victim block in L2 Cache. If there is an Invalid block in the target set, all blocks in the set will be assessed once, Else all blocks in the set will be assessed twice. Due to this overhead, there is significant increase in the execution time. Fig 4.14 shows the comparison of Execution time of proposed algorithm with EqualWrites. The proposed algorithm incurs 23% additional execution time when compared with EqualWrites.

Chapter 5

FUTURE WORK

There is significant raise in the execution time in the proposed algorithm, when compared with EqualWrites algorithm. This is due to the large computation happening during the selection of the victim blocks. If there is Invalid blocks, all the blocks in the set will be involved in the computation once. Else the computation will happen twice for all block sequentially. This increases the execution time, as associative increases in the cache. Instead of Iterative approach, parallel algorithm can be used during the victim block selection. Victim block will be identified in 2 steps during the worst case (No Invalid lines in the target set) and will reduce the execution time to O(1) Linear time complexity instead of O(associative) time complexity.

Chapter 6

CONCLUSION

Considering NVMs having low write endurance and the existing cache management policies are having write variation (WV) among the blocks, new cache replacement algorithm has been proposed for reducing the write variations among the block, for reducing the write error rates and for reducing number of writes into the cell. The proposed algorithm place the incoming block in a line that incurs the minimum error rate and minimum number of writes in Write operation. This is done by comparing the contents of the incoming block with lines in a cache set. Proposed algorithm is implemented using gem5 Simulator and evaluated with PARSEC benchmark suite. The Proposed algorithm is compared with EqualWrites algorithm. Proposed algorithm improves the lifetime of STT-RAM caches by 27%

REFERENCES

[1] Cho, Sangyeun, and Hyunjin Lee., *"Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance."*, Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on. IEEE, 2009.

[2] Mittal, Sparsh, and Jeffrey S. Vetter., *"AYUSH: A technique for extending lifetime of SRAM-NVM hybrid caches"*, IEEE Computer Architecture Letters 14.2 (2015): 115-118.

[3] RJue Wang, Xiangyu Dong, Yuan Xie, Norman P. Jouppi, *" i^2 WAP: Improving Non-Volatile Cache Lifetime by Reducing Inter- and Intra-Set Write Variations"*, Mathematical Methods and Techniques in Engineering and Environmental Science.

[4] Monazzah, Amir Mahdi Hosseini, Hamed Farbeh, and Seyed Ghassem Miremadi., *"LER: Least-error-rate replacement algorithm for emerging STT-RAM caches."*, IEEE Transactions on Device and Materials Reliability 16.2 (2016): 220-226.

[5] Sparsh Mittal, Jeffrey 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, JANUARY 2016

[6] Yang, Byung-Do, et al. *"A low power phase-change random access memory using a data-comparison write scheme."*, Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on. IEEE, 2007.

[7] Bienia, Christian, et al. *"The PARSEC benchmark suite: Characterization and architectural implications."*, Proceedings of the 17th international conference on Parallel architectures and compilation techniques. ACM, 2008.

[8] Binkert, Nathan, et al. *"The gem5 simulator."*, ACM SIGARCH Computer Architecture News 39.2 (2011): 1-7.

[9] https://gem5.org

[10] https://parsec.cs.princeton.edu/