CoPA: Cold Page Awakening to Overcome Retention Failures in STT-MRAM Based I/O Buffers

Mostafa Hadizadeh[1], Elham Cheshmikhani[1], Maysam Rahmanpour[1], Onur Mutlu[2], and Hossein Asadi[1]

Abstract—Performance and reliability are two prominent factors in the design of data storage systems. To achieve higher performance, recently storage system designers use $Dynamic\ RAM$ (DRAM)-based buffers. The volatility of DRAM brings up the possibility of data loss and data inconsistency. Thus, a part of the main storage is conventionally used as the journal area to be able of recovering unflushed data pages in the case of power failure. Moreover, periodically flushing buffered data pages to the main storage is a common mechanism to preserve a high level of reliability. This scheme, however, leads to a considerable increase in storage write traffic, which adversely affects the performance. To address this shortcoming, recent studies offer a small $Non{-}Volatile\ Memory$ (NVM) as the $Persistent\ Journal\ Area$ (PJA) along with DRAM as an efficient approach to overcome DRAM vulnerability against power failure while effectively reducing storage write traffic. This approach, named $NVM{-}Backed\ Buffer$ (NVB-Buffer), features from advantages of NVMs and addresses DRAM shortcomings. In this paper, we employ the most promising technologies for PJA among the emerging technologies, which is $Spin{-}Transfer\ Torque\ Magnetic\ Random\ Access\ Memory$ (STT-MRAM) to meet the requirements of efficient PJA by providing high endurance, non-volatility, and DRAM-like latency. Despite these advantages, STT-MRAM faces major reliability challenges, i.e. *Retention Failure*, *Read Disturbance*, and *Write Failure*, which have *not* been addressed in previously suggested NVB-Buffers. In this paper, we first demonstrate that the retention failure is the dominant source of errors in NVB-Buffers as it suffers from long and unpredictable page idle intervals (i.e., the time interval between two consecutive accesses to a PJA page). Then, we propose a novel NVB-Buffer management scheme, named, $Cold\ Page\ Awakening$ (CoPA), which predictably reduces the idle time of PJA pages. To this aim, CoPA employs $Distant\ Refreshing$ to periodically overwrite the vulnerable PJA page contents by opportunistically using their replica in DRAM-based buffer. We compare CoPA with the state-of-the-art schemes over several well-known storage workloads based on physical journaling. Our evaluations show that CoPA significantly reduces the maximum page idle time, which leads to three orders of magnitude lower failure rate with negligible performance degradation (1.1%) and memory overhead (1.2%).

Index Terms—Data Storage Systems, Persistent Journal Area, STT-MRAM, Retention Failure

1 Introduction

Storage systems use *Dynamic Random Access Memory* (DRAM)-based buffers to mitigate the considerable latency gap between storage and main memory [1–4]. DRAM benefits from low access time, high density, and unlimited lifetime, which makes it a suitable candidate for these applications. However, data loss due to power failure is the major drawback of DRAM-based buffers because of its volatility. Thus, DRAM-based buffers exploit mechanisms such as periodic flush [2, 5] and/or partial use of main storage as the journal area [6–10] to recover data in the case of power failure. Nevertheless, these techniques significantly increase storage write traffic.

Thanks to emerging *Non-Volatile Memories* (NVMs) [11–19], a recent approach to design an efficient buffer is to use a relatively small NVM as the journal area [2, 8] along with DRAM-based buffer. This approach is referred to as *NVM-Backed Buffer* (NVB-Buffer) in this study. In this approach, the buffer takes advantage of NVM persistency to reliably store dirty data pages, beside low latency of DRAM for fast accesses. To this end, NVM is recognized as $Persistent\ Journal\ Area$ (PJA) where a copy of all dirty data pages is stored in NVM. In this case, there always exists a *persistent* valid copy of the DRAM data page. Thus, NVB-Buffer can use PJA data pages for data recovery in the case of power failure while considerably reducing storage write traffic [2].

Previous studies attempt to use *Phase Change Memory* (PCM) technology in NVB-Buffer [9, 20, 21]. PCM, however, suffers from limited endurance, poor write performance, and considerable write power [22–27], which makes it an ineffective candidate for NVB-Buffers. Moreover, existing NVM candidates such as Flash, Ferroelectric RAM (FeRAM), 3D-Xpoint, and Resistive RAM (ReRAM) either suffer from high write latency (3D-XPoint, ReRAM, and Flash) or limited lifetime (3D-XPoint, Flash, and FeRAM) [17, 28–31]. Fig. 1 illustrates the existing memory technologies with respect to PJA requirements. Among these candidates, *Spin-Transfer Torque Magnetic Random Access Memory* (STT-MRAM) technology meets the PJA requirements

1. Mostafa Hadizadeh, Elham Cheshmikhani, Maysam Rahmanpour, and Hossein Asadi (corresponding author) are with the Department of Computer Engineering, Sharif University of Technology, Tehran 11155-11365, Iran. Emails: mhadizadeh@ce.sharif.edu, elham.cheshmikhani@sharif.edu, rahmanpour@ce.sharif.edu, asadi@sharif.edu

2. Onur Mutlu is with the Department of Computer Science, ETH Zurich, Switzerland. Email: omutlu@gmail.com

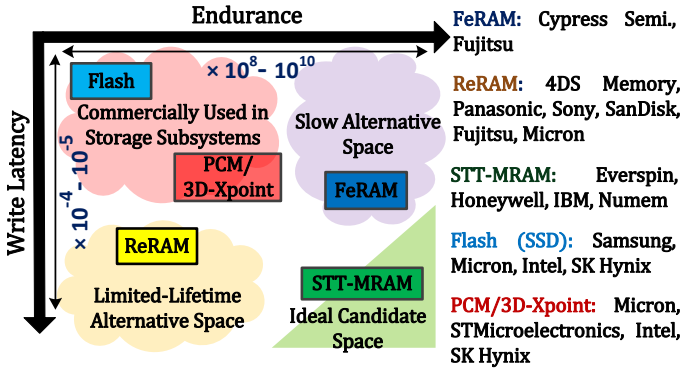


Fig. 1: Overview of existing NVMs with respect to ideal PJA requirements [17, 28–41]

of non-volatility [17], high endurance [32], and DRAM-comparable performance [17], as well as high density and negligible leakage current [33]. Therefore, compared to the other PJA candidates, STT-MRAM is considered as one of the most promising technologies for NVB-Buffers [14, 17, 29–32].

Despite STT-MRAM advantages, it faces three reliability challenges such as *Retention Failure*, *Write Failure*, and *Read Disturbance*. Bit errors in a PJA page due to these challenges lead to data loss in the case of power failure, as there is no valid copy of the page in the main storage. Moreover, previous studies mainly aim to enhance performance or reduce journaling overhead [2, 7, 9, 42]. To our knowledge, *none* of the prior studies considers the reliability of STT-MRAM based NVB-Buffers, while the previous schemes to overcome STT-MRAM retention failure are either expensive or not applicable as they increase the probability of read disturbance or cause performance degradation [45–48].

In this paper, we first investigate existing NVB-Buffer schemes using several well-known storage workloads to explore their vulnerability to STT-MRAM error sources. Our extensive set of experiments on *Microsoft Research Cambridge* (MSRC) traces [49] show that PJA pages in NVB-Buffer suffer from long idle time, varying from 34.5 to 100.8 minutes, which makes them highly vulnerable to retention failure. Moreover, technology downscaling makes retention failure more severe [33, 43]. Based on our experiments, retention failure is the dominant source of PJA failures in recent technology nodes, as the probability of data loss due to retention failure is five orders of magnitude higher than write failure, on average [33, 43]. However, read disturbance is not a serious challenge in PJA as the PJA pages are inherently used for data residency (pages are just read in the case of data recovery due to power failure).

We then propose a novel management scheme for NVB-Buffers, named, *Cold Page Awakening* (CoPA) to efficiently reduce the idle time of PJA pages. CoPA reduces idle intervals using *Distant Refreshing*, where it overwrites PJA pages using error-free replicas in DRAM-based buffers. CoPA also differentiates PJA pages based on their vulnerabilities to retention failure and prevents recently written pages from refreshing. To show the effectiveness of Distant Refreshing, we examine and compare the proposed scheme against the conventional refreshing. Our experiments show that Distant Refreshing reduces the failure rate by 89.9%, on average, compared to conventional refreshing. Conventional refreshing employs read-correct-write approach, which increases the probability of read disturbance and, consequently, leads to increase in total failure rate.

CoPA aims at providing different levels of reliability and can be tuned depending on the application requirements. Our experimental evaluations based on the physical journaling [5, 6] are performed across twelve storage workloads from MSRC [49]. CoPA is tuned to guarantee an upper bound for maximum idle time of PJA pages in such a way that reduces the maximum idle time of PJA pages by an average of 53.5× (up to 66.9×) compared to the state-of-the-art NVB-Buffer management schemes [2, 9, 10]. CoPA results in three orders of magnitude failure rate reduction, with negligible performance (an average of 1.1%) and memory (1.2%) overhead. We also compare CoPA with the conventional periodic flush-enabled scheme, which provides high reliability at the cost of considerable storage write traffic. Tuning CoPA in favor of reliability leads to an average of 43% (up to 81.9%) response time reduction.

The main **contributions** of this paper are as follows:

- This is the *first* study that investigates the reliability of STT-MRAM based NVB-Buffers. We examine the state-of-the-art schemes and show that retention failure is the main contributor to PJA errors. PJA pages suffer from long page idle times, which significantly degrades NVB-Buffer reliability due to high probability of retention failure.
- We propose a novel NVB-Buffer management scheme called CoPA, aims at reducing the idle time of PJA pages to mitigate their vulnerability to retention failure. It overwrites PJA pages by employing their valid replica in DRAM while categorizes PJA pages based on their vulnerability to retention failure and prevents from overwriting of recently-written pages.
- Our evaluations show that CoPA can be tuned according to the application reliability requirements, while considerably improves performance compared to the conventional reliable schemes.
- We extensively evaluate CoPA in terms of failure rate and reliability. The results illustrate that CoPA significantly reduces the error rate by three orders of magnitude with very negligible performance and memory overhead compared to the state-of-the-art schemes (1.1% and 1.2% on average, respectively).

2 Background

2.1 NVM-Backed Buffer

Buffering I/O requests is one of the conventional solutions for performance improvement due to reducing storage write traffic and buffer hits in storage systems [3]. In addition to performance improvement, in high-performance *Solid-State Drive* (SSD)-equipped storage systems, reducing storage write traffic leads to the lifetime improvement, as Flash-based SSDs suffer from limited endurance [50–52]. DRAM-based buffers, however, face with unrecoverable data loss in case of power failure due to volatile characteristic of DRAM. To address this challenge, several techniques such as a) periodically flushing buffered pages to the main storage [2, 5]

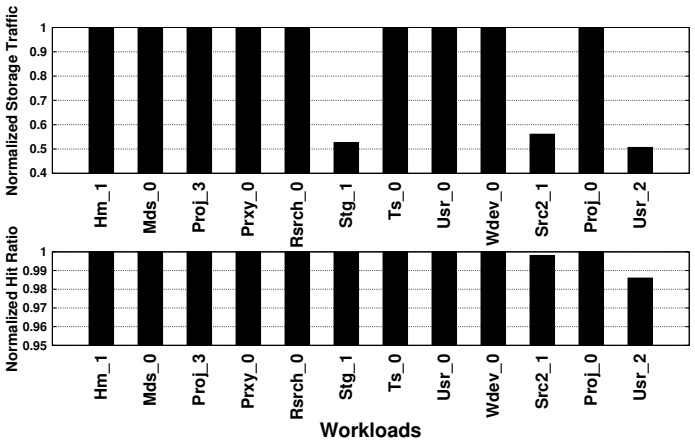


Fig. 2: Hit ratio and storage traffic of NVB-Buffer normalized to Hyb-Buffer

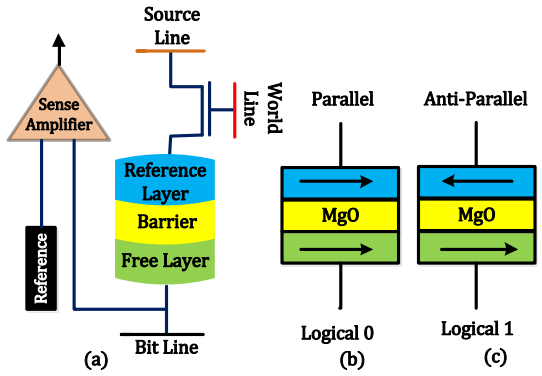


Fig. 3: a) STT-MRAM cell structure, b) parallel state, and c) anti-parallel state [17, 53]

and b) partial use of the main storage as the journal area [6–10] are exploited for data recovery in the case of power failure. Nevertheless, these techniques lead to a considerable increase in storage write traffic [2].

Taking advantage of NVMs persistency along with DRAM performance is one of the promising schemes for designing efficient buffers [1, 2]. *NVM-Backed Buffer* (NVB-Buffer) is one of the recent schemes that provides efficient I/O buffers. In this scheme, DRAM-based buffer is equipped with a relatively small NVM as a *Persistent Journal Area* (PJA), which *only* tracks the dirty data pages of DRAM. In NVB-Buffers, the NVM pages are only read in the case of data recovery while in other scenarios, the accesses to NVM are only writing dirty pages. NVM pages are flushed whether NVM becomes full or their corresponding pages in volatile buffer are evicted. Thus, DRAM pages always have a persistent copy that prevents data loss in case of power failure.

We analyze the efficiency of NVB-Buffer compared to the conventional *One-tier Hybrid Buffer* (Hyb-Buffer) to understand the impact of adding NVM space to operational buffer capacity on system performance. In Hyb-Buffers, DRAM and NVM are managed as a single tier, where the dirty data pages are redirected to both memories to prevent data loss in the case of power failure. Also, evicted data pages from DRAM are admitted to the NVM while NVM hit leads to migration of the page to the DRAM (if the page is dirty, NVM still buffers its copy). Fig. 2 shows the hit ratio and storage traffic of NVB-Buffer normalized to the Hyb-Buffer. The evaluations are performed based on a buffer consists of 8GB DRAM and 512MB NVM. Although both schemes provide the same performance behavior in workloads with working set smaller than the buffer capacity, NVB-Buffer significantly reduces storage traffic in workloads with working set greater than the buffer capacity. NVB-Buffer alleviates storage traffic by 47.4%, 43.9%, and 49.4% in Stg_1, Src2_1, and Usr_2, respectively. Hyb-Buffer, however, provides slightly higher hit ratio in the same set of workloads (up to 1.4%).

The traffic reduction is achieved by providing more presence time for dirty data pages in the buffer. Inserting the evicted data pages from DRAM to the NVM increases the number of NVM evictions. If the evicted page from NVM is dirty, as a DRAM dirty page should have a persistent copy, the page is written to the main storage and the status of its corresponding page in DRAM is changed to clean. Thus, even if the page still resides in the buffer, it is already flushed to the main storage. We also evaluate the scenario where NVM tracks dirty pages even after the eviction of their corresponding copy from DRAM to provide less storage traffic and more chance for buffer hit. The improvement of this scenario, however, is less than 0.01%, at the best case.

2.2 STT-MRAM Basics

The most promising technology among emerging NVMs is STT-MRAM, which can be used in NVB-Buffers. An STT-MRAM cell consists of an access transistor and a *Magnetic Tunnel Junction* (MTJ), which is responsible for data storage, shown in Fig. 3-a. MTJ is made of an insulating layer (Magnesium Oxide), and two magnetic layers named *Free Layer* and *Reference Layer*. The magnetic field direction of the reference layer is fixed while the direction of the free layer can be changed and determine the constructed resistance value. If the free layer is paralleled with the reference layer in the case of flowing write current (I_{write}) from the free layer to the reference layer, MTJ will be in low resistance state, which is the logical '0' value (Fig. 3-b). An anti-parallel pattern between the spin directions of the free layer and the reference layer is due to I_{write} flowing from the reference layer to the free layer, which results in high MTJ resistance and logical '1' value (Fig. 3-c) [17, 53].

STT-MRAM provides promising features in terms of endurance, performance, power consumption, and scalability. It is expected that STT-MRAM partly replaces technologies such as NOR Flash, *Static RAM* (SRAM), and DRAM, while it is estimated that standalone *Magnetic RAM* (MRAM) and STT-MRAM baseline revenues reach $3.8B in 2029 [54]. However, STT-MRAM is threatened by three types of failures. To read the stored data in an STT-MRAM cell, a small current is applied to the free layer to recognize *high* or *low* resistance level. It is probable that this small current leads to a bit flip in the cell, which is called *Read Disturbance*. To write data, a write current is applied to change the magnetic field

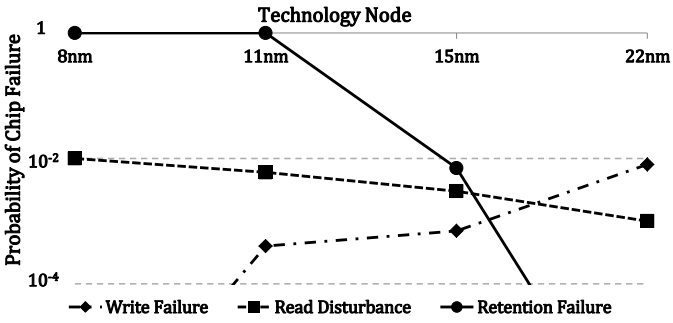


Fig. 4: Footprint of each error type on the probability of STT-MRAM failure [43, 44]

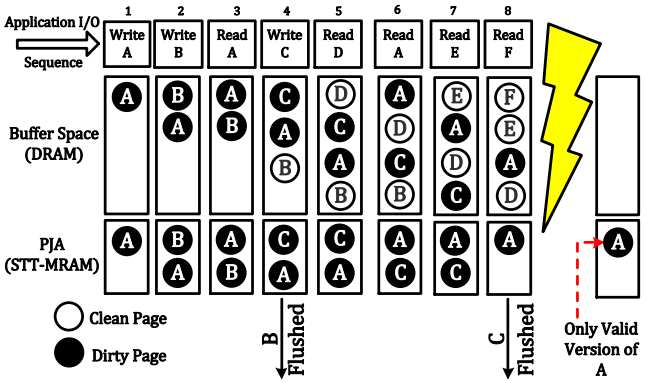


Fig. 5: An example of NVB-Buffer access pattern impact on PJA page idleness

direction of the free layer. Nevertheless, the cell content may not change during the write pulse, which leads to a *Write Failure*. Moreover, stochastic characteristics of STT-MRAM leads to inadvertent bit flip, as the stored data would be changed, without applying any current to the MTJ, and leads to a *Retention Failure*. To make STT-MRAM a widely commercialized replacement for conventional technologies to be used in NVB-Buffers, it is a must to address its reliability challenges.

3 Motivation

3.1 Reliability Challenges of STT-MRAM Based PJA

Any candidate technology to be used in PJA needs to provide three features, a) **non-volatility**: PJA must be capable of retaining data even in the case of power failure to be used for data recovery, b) **high endurance**: due to extreme write pressure on PJA and its relatively small size (compared to DRAM), it should be able to endure a high number of writes without lifetime degradation, and c) **DRAM-comparable write latency**: to prevent inconsistency in reading data from PJA and the buffer (because of large difference between their write performance), PJA delay should be less or equal to DRAM buffers.

Among emerging technologies, STT-MRAM is the most suitable for PJA. PCM, 3D-Xpoint, and Flash neither can provide PJA endurance nor its performance requirements [28–31]. Although FeRAM offers high endurance, its write latency is considerably higher than STT-MRAM [17, 29, 30]. ReRAM is another possible PJA candidate, which provides promising performance feature. ReRAM-based PJA suffers from limited lifetime [17, 29, 30, 32]. Therefore, due to STT-MRAM high endurance and low write latency (compared to FeRAM, PCM, Flash, and 3D-Xpoint), it is the most promising PJA candidate. On the other hand, STT-MRAM provides lower write energy compared to the mentioned technologies [30]. However, to provide an efficient NVB-Buffer, the aforementioned reliability challenges of STT-MRAM should be carefully addressed.

Fig. 4 shows the footprint of each error type on the total failure rate of a 32MB STT-MRAM cache [43]. The probability of write failure is considerably reduced by technology downsizing, while the probability of read disturbance is increased. Although read disturbance is still a serious concern in processor cache levels and main memory [45, 56, 57], from the PJA perspective, read disturbance is not the main challenge as PJA is inherently exploited for data residency. Thus, the stored data in PJA is only read for the sake of data recovery due to power failure or system crash (once during its lifetime). By technology node downsizing, however, retention failure becomes the dominant source of STT-MRAM errors [13, 43, 44, 58] and leads to a significant reduction in STT-MRAM *Mean Time To Failure* (MTTF), as shown in Fig. 4 [33]. Therefore, PJA data pages could only be valid for a very short time intervals. Increasing the probability of retention failure is one of the main challenges of employing STT-MRAM as PJA, since it threatens data retaining, which is essential for PJA.

3.2 Impact of NVB-Buffer Accesses on PJA Page Idle Time

To examine the idle time of PJA pages, which is the main contributor of retention failure rate, a motivational example of how NVB-Buffer access pattern affecting the PJA page idleness is provided as shown in Fig. 5. It is assumed that the buffer and PJA capacity are four and two pages (for the sake of visibility), respectively, while both use *Least Recently Used* (LRU) replacement policy and physical journaling [6] is employed. 1st and 2nd accesses lead to the insertion of *A* and *B* in the buffer and PJA (as both requests are write-requests). The 3rd access leads to reading *A*, which calls to update both queues. However, *A* is read from the buffer and the page *A* in PJA is not either read or refreshed, so it is still idle in PJA. At 4th access, *C* is written, which leads to eviction of *B* from PJA. Thus, *B* is written to the main storage and interpreted as a clean page.

From the 5th access to the 7th access, the buffer is filled, *A* is redirected to the *Most Recently Used* (MRU) position, and *B* is evicted from NVB-Buffer. In the 8th access, *F* is inserted into the buffer leading to the eviction of *C* from the buffer. As *C* is a dirty page, it is redirected to the main storage and its corresponding page is also evicted from PJA. *A* is admitted to the NVB-Buffer at the 1st access and then is read multiple times, which prevents *A* from redirecting to the main storage. In the case of system crash or power failure, the only dirty page that should be recovered is *A*, which is exposed to retention failure due to its long idle time.

This instance demonstrates the access sequences on a small scale. As it shows, a long idle time can be caused for a page after a while, during the access sequences. In this case, long durations should be shortened to decrease the retention failure rate. Therefore, a reliable NVB-Buffer management scheme should seriously consider mechanisms to alleviate the vulnerability of pages such as *A*.

3.3 Impact of Various Error Types on PJA Failures

We provide an illustration about the contribution of retention failure in PJA failures compared to other reliability challenges. We set up an experiment to investigate the impact of each error type on the failure of PJA pages. NVB-Buffer consists of 8GB DRAM and 512MB STT-MRAM based PJA, while employing physical journaling with 4KB page size.

3.3.1 Formulation

Retention failure depends on the idle intervals and the probability of its occurrence for a cell is according to (1) [43]:

$$P_{rf_{cell}}(t) = 1 - e^{\frac{-t}{e^{\Delta}}}, \quad (1)$$

where t and Δ represent idle time and thermal stability factor, respectively. Suppose that each STT-MRAM page consists of 512 64-bit word, and each word is protected by a *Single Error Correction-Double Error Detection* (SEC-DED) code. The probability of data loss due to retention failure of a page for idle time equal to *t* is according to (2):

$$P_{dl\_rf_{page}}(t) = 1 - [(1 - P_{rf_{cell}}(t))^k + k \times (1 - P_{rf_{cell}}(t))^{k-1} \times (P_{rf_{cell}}(t))]^W, \quad (2)$$

where *k* is the number of bits in a word and *W* is the number of words in a page, which are 64 and 512, respectively.

The probability of data loss of a page for all idle intervals is according to (3):

$$P_{dl\_rf_{all\_intervals}} = 1 - \prod_{i=1}^{n}(1 - P_{dl\_rf_{page}}(t_i)), \quad (3)$$

where n is the number of intervals while t_i represents the page idle time during interval *i*.

Write failure is another error type that is likely to happen in PJA pages. The probability of write failure for an STT-MRAM cell is according to (4) :

$$P_{wf_{cell}} = e^{\frac{-t_{write} \times 2 \times \mu_\beta \times p \times (I_{write} - I_{Co})}{c + (e \times m \times (1+p^2)) \times ln(\pi^2 \times \Delta/4)}}, \quad (4)$$

where t_{write} represents the width of write pulse, μ_β is *Bohr* magneton, p is the tunneling spin polarization, and I_{write} represents write pulse width, while c and m are *Euler* constant and magnetic momentum of the free layer, respectively.

The probability of data loss due to write failure for a SEC-DED protected page is according to (5):

$$P_{dl\_wf_{page}} = 1 - [[(1 - P_{wf_{cell}})^k + k \times (1 - P_{wf_{cell}})^{k-1} \times (P_{wf_{cell}})]^W]^N, \quad (5)$$

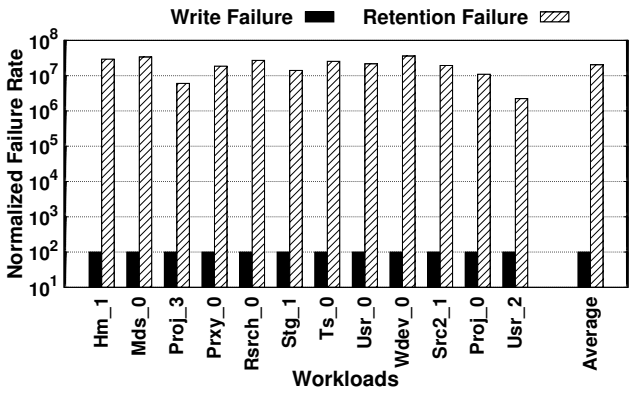


Fig. 6: Impact of each STT-MRAM error type on PJA failures

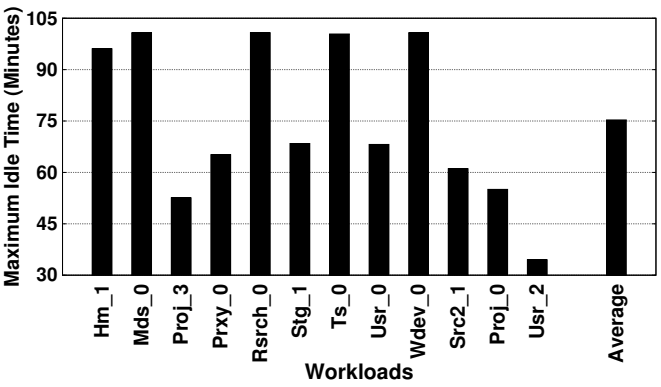


Fig. 7: Maximum page idle time of PJA

where *k* is the number of bits in a word, *W* is the number of words in a cell, and *N* represents the number of writes committed to a page. In NVB-Buffer, since PJA content is just read once for data recovery upon power failure, the impact of read disturbance on PJA failure is highly negligible.

3.3.2 PJA Failure Characterization

To investigate the probability of PJA data loss due to each STT-MRAM error type in recent technology nodes, we assume the probability of an erroneous write is 10^{-8} for an STT-MRAM cell [33, 43, 57]. The probability of data loss for each error type is calculated based on equations (1)-(5). Note that read disturbance is not a major concern in PJA as it is only probable in case of data recovery where each PJA page is read once.

Fig. 6 shows the probability of PJA data loss, normalized to data loss due to write failure. Retention failure is the dominant source of failure as its footprint is drastically higher than write failure (an average of five orders of magnitude). Besides technology node characteristics, long and undetermined idle time of PJA pages is one of the main contributors to the data failure due to retention failure. Fig. 7 shows the maximum page idle time of PJA pages. Neglecting page idleness as one of the design parameters of NVB-Buffer management scheme leads to pages with maximum idle time for an average of 75 minutes (up to 100.8 minutes), which makes them more vulnerable to retention failure.

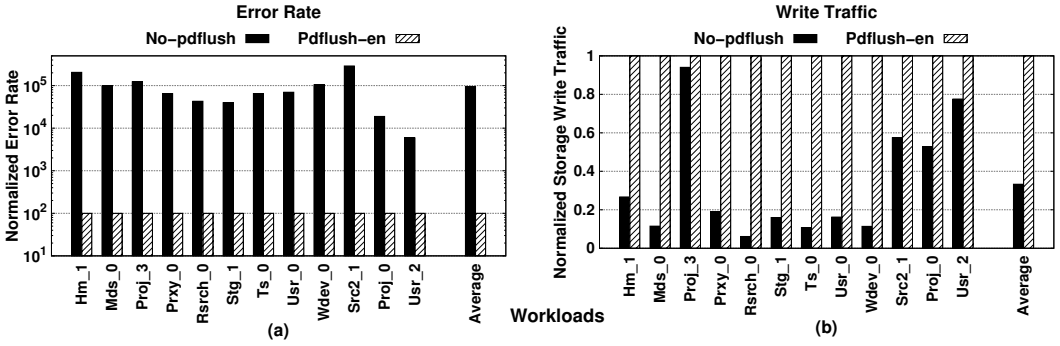


Fig. 8: Impact of periodic flushing on a) error rate and b) storage write traffic: No-pdflush (eliminated periodic flush) vs. Pdflush-en

One of the conventional approaches that can reduce page idle time in PJA is periodic flush, the same approach as Linux *pdflush* function. In this approach, the dirty pages of NVB-Buffer are flushed to the main storage, based on a predetermined time period. Using periodic flush, an upper bound is set for maximum idle time of PJA pages as data is written to the main storage and PJA data pages can be discarded. Despite reducing page idle time, employing periodic flush considerably increases the storage write traffic [2]. We extend our experiment to show the correlation between the maximum idle time of pages and the error rate while investigating the performance impact of the state-of-the-art scheme and a scheme inspired by the approach used in the conventional buffers.

Fig. 8 shows the impact of *using* (Pdflush-en) and *eliminating* periodic flush (No-pdflush) on PJA reliability and storage write traffic. In Pdflush-en, dirty pages of NVB-Buffer are monitored and the flushing procedure is invoked every five seconds to flush the pages with 30-seconds idle time. Enabling periodic flush in Pdflush-en reduces the failure rate by three orders of magnitude (940×, on average) compared to No-pdflush by preventing long page idle time and guaranteeing an upper bound for data pages (depicted in Fig. 8-a). Nevertheless, employing Pdflush-en significantly increases storage write traffic (as shown in Fig. 8-b). Although using No-pdflush increases the probability of data loss, it reduces storage write traffic by an average of 66.7% (up to 93.9% in Rsrch_0), as it provides more chance of re-accessing for dirty data pages in NVB-Buffer.

Designing an efficient NVB-Buffer needs approaches that address long idle time of PJA pages with respect to the storage write traffic. However, *none* of the prior studies address the reliability challenges of STT-MRAM based PJA. Moreover, existing schemes for mitigating retention failure at circuit- and/or architecture-level are either expensive or inefficient and optimized for processor cache or main memory [45, 60]. Based on these observations, we investigate that there is a need for a system-level scheme to enhance the reliability of STT-MRAM based NVB-Buffers, which is the motivation of this work.

4 Proposed Scheme

A commercialized STT-MRAM based NVB-Buffer needs to reconcile its retention failure challenge. To this end, we propose a novel NVB-Buffer management scheme named *Cold Page Awakening* (CoPA) to reduce the idle time of PJA pages. Fig. 9 shows the overall structure of CoPA. While the buffer regularly performs its conventional functions such as a) admission of read/write requests to NVB-Buffer, b) handling buffer hits, and c) metadata management, it also aims at preventing retention failure using NVB-Buffer characteristics. CoPA employs *Distant Refreshing* instead of conventional refreshing [46] mechanism to overcome the large idle time of PJA pages. Regarding the features of recent technologies and NVB-Buffer structure, Distant Refreshing aims at refreshing PJA pages based on their corresponding replica in buffer space without increasing the probability of read disturbance.

CoPA controls the refreshing procedure by tracking the metadata of PJA pages using two queues and a simple 2-bits counter (bits are indicated as QI and DC in Fig. 9) to differentiate between recently-written and idle pages. Consequently, CoPA reduces the probability of retention failure by time-based refreshing of idle pages, without intensifying the probability of read disturbance. The rest of this section details Distant Refreshing mechanism and its benefits over conventional refreshing (Section 4.1), then provides a step-by-step depiction of page awakening procedure (Section 4.2), and finally presents the flow of CoPA for each request (Section 4.3).

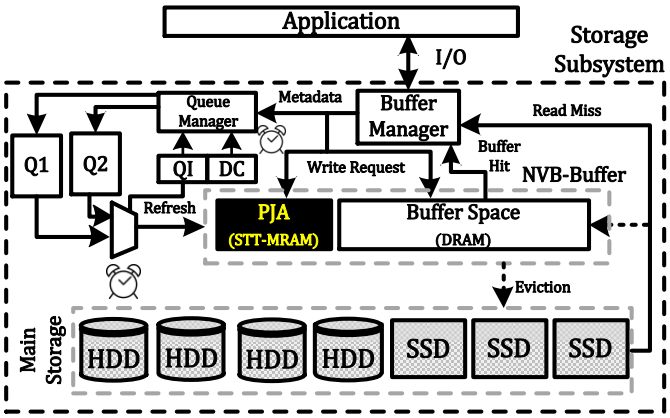


Fig. 9: Overview of CoPA (QI: Queue Identifier, DC: Drowsiness Categorizer)

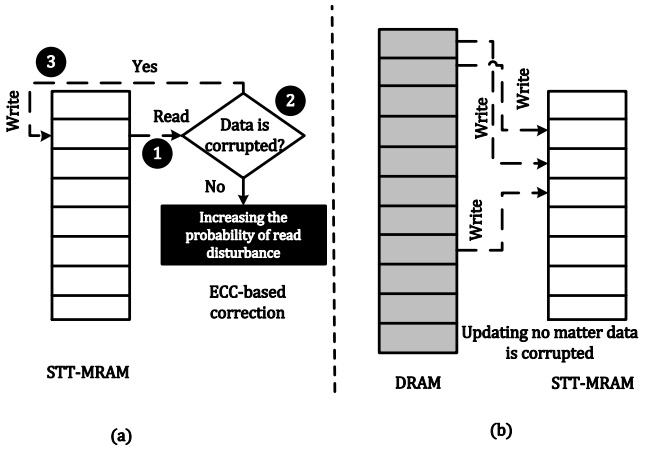


Fig. 10: Refresh approaches: a) conventional refreshing, and b) distant refreshing

4.1 Distant Refreshing

Previous schemes use the conventional refreshing approaches to overcome retention failure of STT-MRAM as shown in Fig. 10-a, where the content of a page is read in step 1, corrected by ECC checking in step 2, and re-written in the case of corruption in step 3. This procedure can be performed based on: 1) on-read checking, 2) time intervals, or 3) prediction [45–48]. Conventional refreshing is efficient when using stand-alone STT-MRAM such as processor caches or main memories. Employing STT-MRAM for PJA, however, faces several challenges. The main challenge of PJA is that it *only* faces with the write requests (except for data recovery procedure), which prevents the on-read checking. Moreover, reading the content of STT-MRAM cells (based on time intervals or prediction) increases the probability of read disturbance, especially on ECC bits, which could lead to uncorrectable errors and data failure. Likewise, using stronger ECCs [45, 55] also faces the mentioned challenges as the strong-ECC bits are vulnerable to read disturbance and retention failure and they impose a considerable memory and performance overhead.

To prevent the negative impact of conventional refreshing on PJA failures (due to the probable read disturbance), we evaluate an NVB-Buffer-friendly approach named *Distant Refreshing* (illustrated in Fig. 10-b), which aims to refresh STT-MRAM cells no matter they are corrupted or not. To refresh an STT-MRAM page in this approach, its corresponding replica in DRAM is read and written in the corresponding STT-MRAM page. Distant Refreshing does not add extra read operations to the STT-MRAM cells as it does not require checking the content of the cells before refreshing. Besides, Distant Refreshing is an appropriate approach for NVB-Buffers as a valid version of each PJA page exists in the DRAM-based buffer space. To investigate the impact of Distant and conventional refreshing on PJA failures, we set up an experiment where for an 8GB buffer with 512MB PJA, each PJA page with idle time equal to 90 seconds is considered for refreshing using both mentioned approaches. The probability of read disturbance for a cell is set to 10^{-7}.

Fig. 11 shows the failure rate of each approach normalized to Distant Refreshing scheme. Conventional refreshing increases failure rate by an average of 9.93× compared to the Distant Refreshing. Distant Refreshing does *not* increase the probability of read disturbance as it does not impose extra reads to PJA. Although Distant Refreshing is threatened by write failure, the probability of write failure is significantly smaller than that of retention failure in recent technology nodes [33, 43, 57]. Thus, it is more beneficial to re-write a page with long idle time and taking the low risk of write failure. Nevertheless, refreshing a recently-written page is inefficient as it has lower idle time. Applying Distant Refreshing to NVB-Buffer does not need extra memory space for storing redundant version of data pages as there is a replica of each PJA page in the DRAM-based buffer space. Hence, this approach is an efficient solution for reducing the idle time of PJA pages and increasing NVB reliability.

4.2 Page Awakening Procedure

4.2.1 Refreshing Aggression

NVB-Buffer can provide a valid replica of each PJA page, which is necessary for Distant Refreshing. However, employing Distant Refreshing faces several challenges such as the length of refreshing intervals (*Refresh Period*) or the way to select some pages for refreshing. In an aggressive approach, all PJA pages are refreshed with a short refresh period. This approach suffers from performance overhead and higher probability of write failure (because of successive write operations). Short refresh period along with refreshing all of PJA pages lead to reading a considerable amount of data from DRAM and writing them to STT-MRAM. Moreover, re-writing pages every few seconds in the aggressive refreshing method can increase the total data failure probability, as it imposes more write failure. Aggressive refreshing also increases the temperature due to extensively reading from DRAM and writing to STT-MRAM, which increases the total probability of failure [33]. Therefore, aggressive refreshing is *not* an efficient approach for NVB-Buffers due to not only its reliability side effects, but also imposing significant memory overhead.

In a conservative refresh approach, the refresh period takes longer, while specific pages are refreshed. However,

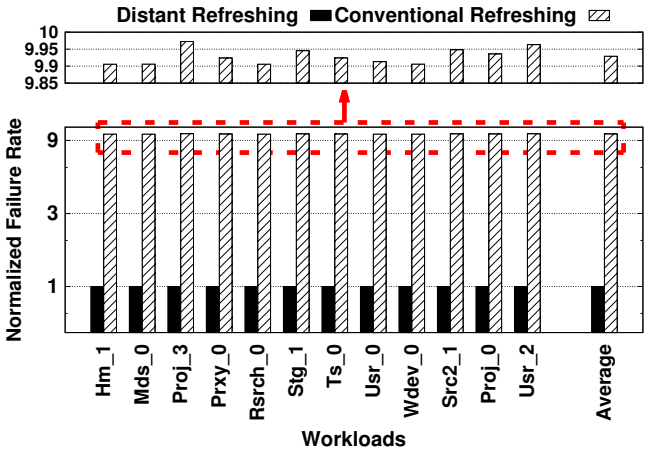


Fig. 11: Impact of distant refreshing vs. conventional refreshing on failure rate

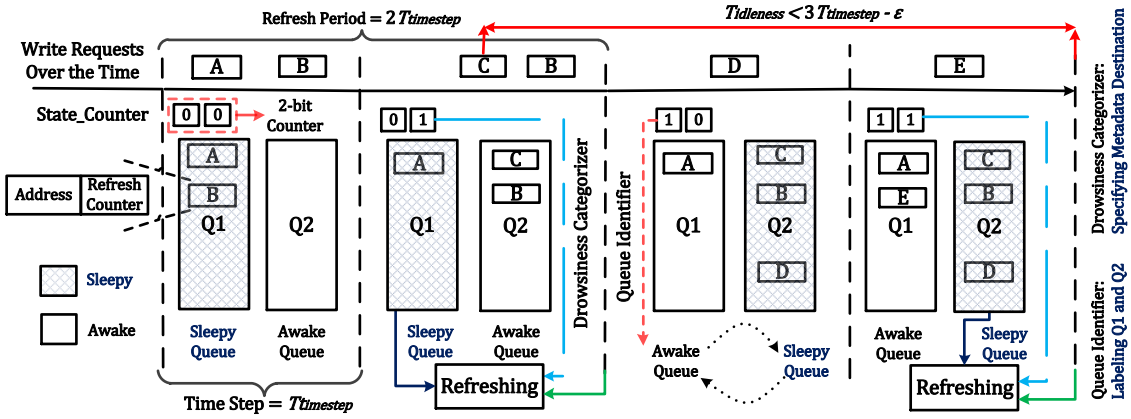


Fig. 12: Queue management in CoPA

a long refresh period increases the probability of retention failure. Moreover, detecting proper pages for refreshing is another challenge of the conservative approach. A simple way to detect the best candidate is continuously checking the access time of each PJA page, which imposes memory overhead. Another solution is to employ detection mechanism such as a) reading the page content and detecting failures by ECC checking, b) comparing the page content with its corresponding copy in DRAM, or c) prediction [45, 46]. However, these solutions *either* cause extra read operations from PJA *or* are not accurate. Thus, conservative refreshing is inefficient due to high memory overhead and still probable retention failure. The ideal scheme should balance between these two extreme approaches.

4.2.2 CoPA Queue Management

CoPA a) increases the length of refresh intervals, b) guarantees an upper bound for idle time of each page, and c) prevents the recently-written pages from refreshing to reduce the overheads. To this end, CoPA takes advantage of two queues, named Q1 and Q2, and interprets them as *Sleepy Queue* and *Awake Queue* to differentiate and manage idle and recent pages. Sleepy and Awake queues are labels that CoPA assigns to Q1 and Q2, determining the page idleness of Q1 and Q2 during the runtime of the workloads, respectively. Sleepy queue is the label of the queue that tracks pages with *high* idle times while Awake queue is the label of the queue that tracks the metadata of pages with *low* idle times. CoPA also uses a 2-bit counter called *State_Counter* to manage both queues, as shown in Fig. 12. *State_Counter* is employed for differentiating and managing Q1 and Q2.

CoPA divides each refresh period into two time-steps where at the end of each time-step, the value of *State_Counter* is increased by one. The *Most Significant Bit* (MSB) and *Least Significant Bit* (LSB) of *State_Counter* are recognized as *Queue Identifier* and *Drowsiness Categorizer*. Fig. 12 shows how CoPA manages these queues, based on the value of *State_Counter*. *Queue Identifier* is responsible for identifying the Sleepy and Awake queues. When *Queue Identifier* is '0', Q1 is recognized as the Sleepy queue, otherwise, Q2 is interpreted as the Sleepy queue. *Drowsiness Categorizer* manages the insertion of each page metadata. If *Drowsiness Categorizer* is '0', the metadata of incoming page is inserted into the Sleepy queue. On the other hand, the metadata of incoming page is inserted into the Awake queue. The refresh operation is performed at the end of the refresh period (after two time-steps), which leads to *Queue Identifier* transition (from '0' to '1' or from '1' to '0') and refreshing PJA pages based on the Sleepy queue. Thus, CoPA gets rid of the high overhead of tracking and checking the idle time of each page by employing a simple *State_Counter*. Moreover, using *Drowsiness Categorizer*, CoPA is able to prevent refreshing of recently written pages and using *Queue Identifier*, CoPA just changes the label of its queues (Q1 and Q2), instead of copying their contents.

Fig. 12 shows an example of how CoPA manages the refreshing procedure. It is assumed that the initial value of *State_Counter* is 0. In the first time-step, *Drowsiness Categorizer* is '0' so *A* and *B* as incoming requests are inserted into the Sleepy queue. As *Queue Identifier* is '0', Q1 and Q2 are interpreted as the Sleepy and Awake queues, respectively. At the end of each time-step, the value of *State_Counter* is increased by one. In the second time-step, as *Drowsiness Categorizer* is '1', the metadata of requests are redirected to the Awake queue. Since *Queue Identifier* value is not changed, Q1 and Q2 are still the Sleepy and Awake queues, respectively. The metadata of *C* is inserted into Q2 and for *B*, the old version in Q1 is invalidated and then the metadata of *B* is inserted into Q2.

At the end of the second time-step, the refresh procedure is performed based on entries of the Sleepy queue, which is Q1. Thus, the corresponding page of *A* in PJA is refreshed. By increasing *State_Counter* value, the *Queue Identifier* value is changed from '0' to '1'. Hence, the labels of Q1 and Q2 are interchanged, i.e., Q1 becomes Awake queue and Q2 is now the Sleepy queue. As *Drowsiness Categorizer* is '0', the metadata of *D* is inserted into the Sleepy queue (Q2). At the fourth time-step, based on *Drowsiness Categorizer* value ('1'), *E*'s metadata is inserted into the Awake queue (Q1). At the end of the fourth time-step, the refreshing of *C*, *B*, and *D* is performed as their addresses reside in the Sleepy queue (Q2). At the worst case, each page is refreshed after three time-steps. Therefore, if a page is not re-written by the application, the maximum idle time of each page is according to (6):

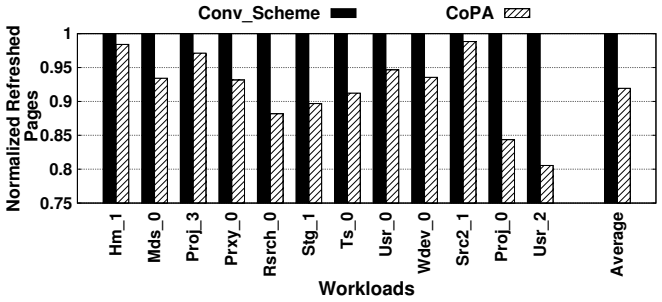


Fig. 13: Number of refreshed pages

$$T_{time-step} + \epsilon < T_{idleness} < 3T_{time-step} - \epsilon, \quad (6)$$

where $T_{timestep}$ is the time-step period (the half of each refresh period), $T_{idleness}$ is the page idleness time, and ϵ is the minimum time unit.

We evaluate CoPA to compare its impact on the number of refreshed pages against an approach with aggressive refreshing named *Conv_Scheme*. *Conv_Scheme* aims to refresh PJA pages by a predetermined period equal to 60 seconds, while CoPA is tuned in such a way that provides an error rate of the same order compared to *Conv_Scheme*. Fig. 13 shows the number of refreshed pages normalized to *Conv_Scheme*. CoPA reduces the number of refreshed pages up to 19.5% compared to *Conv_Scheme*, as it prevents the refreshing of recently written pages. By reducing the refreshing period of *Conv_Scheme*, the gap between CoPA and *Conv_Scheme* is increased.

Regarding the advances in technology, CoPA aims to address the raised vulnerabilities of STT-MRAM based NVB-Buffers while avoiding the degradation of the performance. The conventional approaches do not consider the opportunities provided by NVB-Buffers and preserves the reliability requirements by extensive storage admissions. On the other hand, CoPA proposes a novel solution based on the structure of NVB-Buffers. CoPA does not rely on storage subsystem to provide persistency for dirty pages. Instead, it takes advantage of the NVM in the structure of NVB-Buffer. Accordingly, CoPA avoids the increase in the storage write traffic and offers an NVB-Buffer aware solution to mitigate the page idle time. By employing CoPA, an appropriate level of reliability is achieved while a satisfactory level of the performance is preserved. Thus, CoPA would be an efficient solution for NVB-Buffer management.

4.3 Putting It All Together

Algorithm 1 shows the flow of CoPA scheme. CoPA mainly consists of two procedures: 1) $Req\_Man\text{-}agement$ and 2) $PJA\_\ Refreshing$. *State_Counter* is a 2-bit counter, which is used to distinguish between the Sleepy queue and Awake queue, based on its MSB (QI) and LSB (DC) values. Each entry of these queues consists of the page address of the inserted page.

$PJA\_\ Refreshing$ is invoked based on the time-steps by a timer interrupt. Based on the value of DC, CoPA aims to refresh PJA pages. In $PJA\_\ Refreshing$, the current value of DC is checked to determine that which queue is responsible for tracking incoming requests, and if it is '1', it means that the recent write requests are inserted into the Awake queue and now it is the time for QI transition. Hence, CoPA refreshes PJA pages based on the addresses of the Sleepy queue. Every time $PJA\_\ Refreshing$ is invoked, the value of *State_Counter* is increased by one, which specifies the state of CoPA queues for the next time-step.

Algorithm 1 Procedure of Page Awakening

```
State_Counter: 2-bits
Q1, Q2: CoPA Queues
Def. QI: MSB of State_Counter        //Queue Identifier
Def. DC: LSB of State_Counter        //Drowsiness Categorizer
Def. Q1 is Sleepy_Queue & Q2 is Awake_Queue if QI == 0
Def. Q2 is Sleepy_Queue & Q1 is Awake_Queue if QI == 1
Initial State_Counter=0,

Procedure PJA_Refreshing
begin
    if DC = 1 then
        // Refreshing Operation
        Refresh PJA Pages, Based on Sleepy_Queue
    end
    State_Counter = State_Counter + 1
end

Procedure Req_Management()
begin
    for Each new request P do
        Insert P in NVB_Buffer
        if P is write request then
            Invalid P.adrr If It Was Existed in Q1 or Q2
            if DC = 0 then
                //Sleepy_Queue insertion
                Insert P.addr in Sleepy_Queue
            else
                //Awake_Queue Insertion
                Insert P.addr P in Awake_Queue
            end
        end
        if Dirty page E is evicted from NVB-Buffer then
            Invalid E.adrr If It Was Existed in Q1 or Q2
        end
    end
end
```

5 Evaluation

CoPA also updates its queues by the incoming new write requests. If the page already exists in either of the CoPA

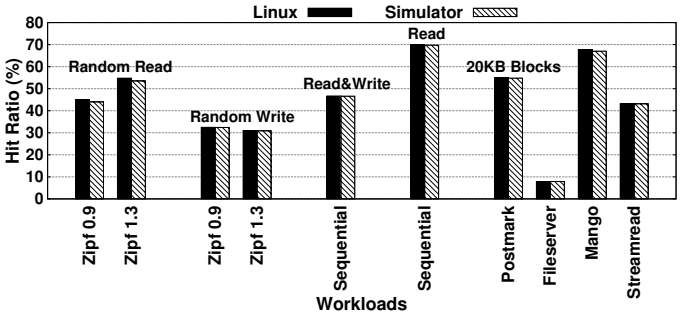


Fig. 14: Accuracy of the buffer cache simulator used for evaluations

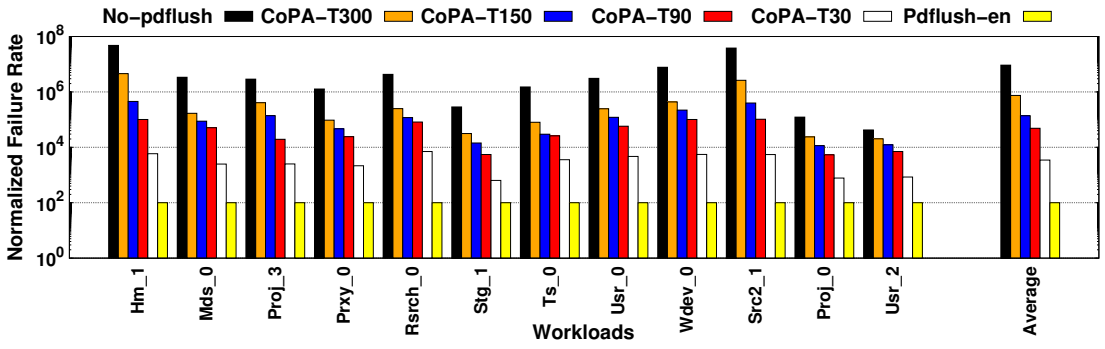


Fig. 15: Failure rate of CoPA (with different time-steps) compared to No-pdflush

queues, the corresponding page is invalidated. Based on the value of QI and DC, CoPA inserts the metadata of this page into the proper queue (into the Sleepy queue if DC is '0' and into the Awake queue, otherwise). Moreover, if a dirty page is evicted from NVB-Buffer, CoPA invalidates its metadata in CoPA queues.

5.1 Evaluation Flow

To evaluate CoPA, we take advantage of our in-house buffer cache simulator, which is developed based on Linux kernel 4.4 and EXT2 filesystem, validated using Flexible I/O (fio) [61], Filebench [62] and Postmark [63] over different types of workloads. As shown in Fig. 14, there is a negligible difference (up to 2.3%) between the simulator and Linux hit ratio under different patterns of accesses (random or sequential) and various distributions (e.g., zipf 0.9 and zipf 1.3), which is due to the difference between employed LRU and the semi-LRU used by Linux. We extend the simulator for: 1) analyzing timing parameters of the application block requests and 2) investigating the impact of each scheme on the response time. The simulator is also equipped with DRAMSim2 [65], configured for simulating two DDR3 modules based on DRAM and STT-MRAM characteristics for the buffer and PJA, respectively [64]. We post-process the simulator outputs based on Equation (1)-(5) to evaluate each scheme in term of retention failure probability.

Our evaluations are performed on twelve workloads from *Microsoft Research Cambridge* (MSRC) traces [49], based on an NVB-Buffer consists of 8GB DRAM as buffer space, 512MB STT-MRAM as PJA, and the page size of 4KB. Each of these pages consists of 512 64-bit data-words, each of which is protected by SEC-DED(64,72) code (an 8-bit redundancy is used for each data-word). CoPA is compared with the the state-of-the-art approaches [2, 9, 10], where the periodic flushing is eliminated, referred to as *No-pdflush* (the physical journaling is considered as the journaling mechanism). To provide an overview of CoPA efficiency in term of performance, we also compare CoPA with a conventional reliable scheme named *Pdflush-en*. Pdflush-en uses periodic flush based on 5-seconds intervals, where pages with 30 seconds idle time are flushed to the main storage and discarded from PJA, which is the default configuration in operating systems such as Linux [2]. Thus, this scheme reduces the probability of retention failures by providing a small idle time for PJA pages, but imposes significant storage write traffic compared to No-pdflush (as shown in Section 3.3.2).

5.2 Failure Rate Analysis

To investigate CoPA in term of reliability, we evaluate it based on different refresh periods and set the time-steps equal to 30 seconds (CoPA-T30), 90 seconds (CoPA-T90), 150 seconds (CoPA-T150), and 300 seconds (CoPA-T300). Fig. 15 shows the failure rate of CoPA and No-pdflush normalized to Pdflush-en. On average, CoPA reduces the probability of failure by three orders of magnitude compared to No-pdflush. CoPA achieved such reduction without increasing the storage write traffic. Pdflush-en imposes extra writes to the storage subsystem to preserve data consistency. Such approach drastically affects the overall performance of the system and it contradicts the primary goals of systems equipped with NVB-Buffers. On the other hand, CoPA tries to exploit the non-volatility provided by NVB-Buffer to eliminate the need for storage accesses and preserve the data consistency. Therefore, CoPA is a viable solution for NVB-Buffer management.

CoPA provides a significant idle time reduction for PJA pages (up to 66.9×) by guaranteeing to refresh each PJA page at least every three time-steps, as shown in Fig. 16. It also provides a predictable PJA idle time to fill the gap between No-pdflush and Pdflush-en approach in term of reliability, as there is no upper bound for PJA page idle time in No-pdflush scheme.

There is a considerable difference between the maximum page idle time of different workloads (more than 66 minutes) in No-pdflush, while this value in CoPA differs in few seconds. Therefore, CoPA alleviates the variation of page idleness compared to the schemes that have eliminated periodic flush, and accordingly, provides a reliable scheme without increasing main storage traffic. CoPA can be tuned to provide the same reliability as Pdflush-en. However, in this case, the trade-off between the number of refreshed pages and failure rate period should be considered, as lower refreshing period leads to higher number of refresh operations.

CoPA effectively mitigates the probability of retention failure compared to DF-LRW [2] by reducing the page idleness in workloads with high idle time. The main focus of CoPA is to prevent retention failure. Therefore, in corner cases where the probability of retention failure is low, PJA should be equipped with solutions addressing other failure types to achieve higher MTTF. For example, in an application with extremely high read intensity, the read disturbance

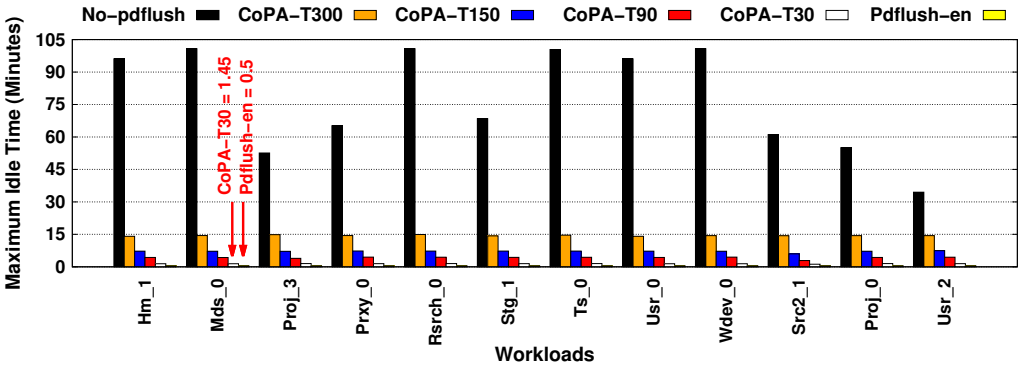


Fig. 16: Maximum idle time of CoPA compared to No-pdflush. Maximum idle time values: *No-pdflush* (100.77 minutes), *CoPA-T300* (14.95 minutes), *CoPA-T150* (7.47 minutes), *CoPA-T90* (4.5 minutes), and *CoPA-T30* (1.5 minutes).

would be the main contributor to PJA failure. In a write-intensive workload, two scenarios can be noted. If the access pattern of the workload is not uniform, then part of PJA pages would remain idle, which are vulnerable to retention failure and CoPA can improve the overall reliability in this scenario. In the scenario where the workload has a uniform access pattern, the idle time of pages is low due to repeated rewrites of pages. In this scenario, a few numbers of idle pages (tracked by Sleepy queue) should be refreshed. So compared to DF-LRW, CoPA roughly affects the write failure in these scenarios. Current experiments have been performed based on an NVB-Buffer equipped with SEC-DED ECC. Thus, in workloads where read disturbance or write failure are the dominant sources of PJA failures, errors are partially recovered by ECC. However, in case the system designers want to take another step to provide more failure rate reduction for these scenarios, they can take advantage of solutions such as [33, 45, 55–57, 59, 73, 74].

5.3 Response Time

One of the key aspects of an efficient NVB-Buffer management scheme is its impact on the response time of storage subsystem. Although Pdflush-en considerably reduces the probability of retention failure, it significantly increases storage write traffic compared to No-pdflush. However, CoPA not only significantly reduces the probability of retention failures compared to No-pdflush, but also avoids Pdflush-en high storage write traffic. CoPA, as well as improving reliability compared to No-pdflush, aims at preventing Pdflush-en poor write traffic.

Fig. 17 shows the response time of each scheme normalized to No-pdflush. The time-step of CoPA is set to 30 seconds, which brings the highest response time (due to higher refreshing operations). CoPA increases response time by an average of 1.1% compared to No-pdflush (up to 4.2%), which is due to extra read and write operations on PJA and buffer. On the other hand, CoPA significantly reduces the response time compared to Pdflush-en. It is noteworthy that Fig. 17 only depicts CoPA-T30 as it obviously has the highest response time between CoPA configurations. Other CoPA configurations provide a response time between CoPA-T30 and No-pdflush. The reliability improvement provided by Pdflush-en is achieved by enabling periodic flushing, which leads to a considerable increase in storage write traffic. CoPA, however, enhances reliability without imposing extra storage write traffic, therefore it can offer a reliable NVB-Buffer management scheme with higher performance compared to Pdflush-en.

Although Pdflush-en provides higher error reduction compared to CoPA, in a system based on NVB-Buffers, employing a scheme with the same approach as Pdflush-en is inefficient. This approach neglects the features provided by NVB-Buffers and significantly increases the storage write traffic. In fact, the main intuition behind equipping systems with NVB-Buffers is to eliminate the costly storage accesses used for preserving data consistency. The error rate of CoPA-T30 is 34$\times$ higher compared to Pdflush-en. However, the average error rate of CoPA-T30 is about 10^{-8} and CoPA is able to provide a proper level of reliability while considerably improves the performance over Pdflush-en. Accordingly, CoPA is a better solution for systems consist of NVB-Buffers.

System designers have the opportunity to get the performance behavior of CoPA closer to No-pdflush, by tuning time-step knob. With proper refresh period, CoPA can provide response time close to No-pdflush without long idle time for PJA pages. However, as already discussed, increasing time-step leads to higher failure rate. Nevertheless, it still can provide an upper bound for PJA pages, and yet higher reliability compared to No-pdflush.

6 Related Work & Discussion

Employing NVMs in buffering schemes has already been addressed in prior studies based on two main approaches: 1) storage-aware approach [4, 66, 67] and 2) storage-unaware approach [2, 7, 9, 16, 42, 68–70]. The first approach aims at improving the lifetime and performance of SSD-based storage systems using different mechanisms such as dirty page favoring [4] (the mechanism of using policies to prioritize dirty pages in the buffer), overcoming small write problem, and reducing garbage collection frequency [52]. These schemes, however, increase the idle time of dirty pages in the NVM, which increase the probability of retention failure.

Schemes in the second approach employ NVMs as part of the main memory [16, 68, 69] or log/journal area [2, 7, 9, 42]. As conventional schemes exploit a part of the main storage (HDD or SSD) as the journal area for crash recovery, this approach takes advantage of NVM persistency

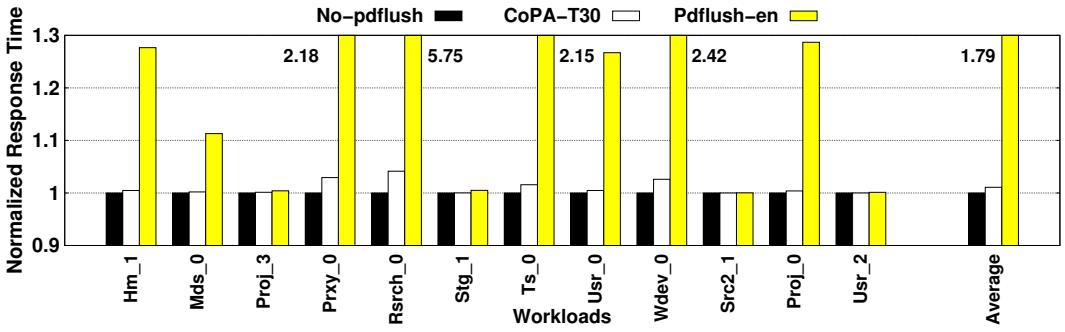


Fig. 17: Normalized response time of No-pdflush, CoPA, and Pdflush-en

to reduce storage traffic by committing journaled data to the NVM. Nevertheless, these schemes are optimized toward performance improvement by decreasing storage write traffic, which is achieved by increasing the residency time of dirty pages in the NVM. Therefore, the schemes in the second approach increase the probability of retention failure compared to the conventional schemes, in favor of reducing storage write traffic.

In [71], mwJFS is proposed to address the retention time of *Multi-Level Cell* (MLC) PCM-based jouranling file systems by employing different write pulse widths (and accordingly retention time). Although expanding the write pulse in order to increase the retention time is a conventional solution in NVMs, and it is also applicable to STT-MRAM, it leads to considerable increase in write latency and power consumption [72]. Moreover, employing the retention monitoring mechanism of mwJFS for STT-MRAM based PJAs increases the probability of read disturbance. CoPA, however, significantly decreases the error rate by reducing the page idle time compared to these schemes without increasing read disturbance probability, while providing a noticeable reduction of storage write traffic compared to the conventional schemes.

CoPA is also applicable to the main memory buffer cache layer. To this end, *Operating System* (OS) needs to assign a small portion of the main memory to the *Awake* and *Sleepy* queues. Employing CoPA for NVB-Buffer management has a negligible memory overhead. The memory overhead results from the queues required by caching and CoPA. For an NVB-Buffer consists of 8GB DRAM and 512MB STT-MRAM, two queues should be allocated for handling the eviction and admission procedure, while CoPA queues also require memory allocation. In a double-link list based implementation, each queue entry consists of three fields, two pointers to the previous and next entries, and one for the page address (same approach as [75]). CoPA queues also can be managed in the same way, while the total size of the Sleepy and Awake queue does not exceed the queue size required for NVM management as they aim to track NVM pages. Assuming 4KB page size and 16B memory access granularity, managing NVB-Buffer using CoPA imposes 1.2% memory overhead to the system. To employ CoPA for systems with logical journaling, a portion of DRAM equal to the size of NVM should be considered to store the required valid copy of journaled data for Distant Refreshing. Logical journaling needs much less space since the updates to the data pages are buffered. Thus, the required DRAM space for Distant Refreshing compared to the total DRAM space is negligible.

7 Conclusion

Employing STT-MRAM as PJA provides the opportunity for designing efficient NVB-Buffer schemes. The technology downscaling increases the probability of retention failure in STT-MRAM, especially in recent technologies where retention failure becomes the main source of errors. Although existing NVB-Buffer management schemes provide considerable storage write traffic reduction compared to the conventional schemes, they suffer from long and undetermined PJA page idle time. The longer the page idle time, the more the error rate compared to the conventional schemes. This paper proposed an efficient scheme named CoPA, for reducing the probability of retention failure. CoPA utilized an NVB-Buffer-friendly approach named *Distant Refreshing*, which re-writes the idle PJA pages based on their replica in DRAM no matter it is corrupted or not. CoPA monitors PJA pages using two queues (Awake and Sleepy queues) to distinguish pages with long idle time. CoPA also guarantees a tunable upper bound for maximum idle time of PJA pages. Our evaluations illustrated that CoPA reduces the probability of failure by three orders of magnitude with negligible performance overhead.

References

[1] Z. Fan, F. Wu, D. Park, J. Diehl, D. Voigt, and D. H. C. Du, "Hibachi: A Cooperative Hybrid Cache with NVRAM and DRAM for Storage Arrays," *in Proceedings of International Symposium on Mass Storage Systems and Technologies (MSST)*, May 2017, pp. 1-12.

[2] E. Lee, H. Kang, H. Bahn, and K. G. Shin, "Eliminating Periodic Flush Overhead of File I/O with Non-Volatile Buffer Cache," *IEEE Transactions on Computers (TC)*, vol. 65, no. 4, pp. 1145-1157, April 2016.

[3] R. Salkhordeh, M. Hadizadeh, and H. Asadi, "An Efficient Hybrid I/O Caching Architecture Using Heterogeneous SSDs," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 30, no. 6, pp. 1238-1250, June 2019.

[4] Z. Fan, D. H. C. Du and D. Voigt, "H-ARC: A Non-volatile Memory Based Cache Policy for Solid State

Drives," *in Proceedings of Symposium on Mass Storage Systems and Technologies (MSST)*, June 2014, pp. 1-11.

[5] V. Prabhakaran, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Analysis and Evolution of Journaling File Systems," *in Proceedings of USENIX Annual Technical Conference (ATC)*, April 2005, pp. 1-16.

[6] L. Lu, Y. Zhang, T. Do, S. Al-Kiswany, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Physical Disentanglement in a Container-Based File System," *in Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, October 2014, pp. 81-96.

[7] E. Lee, H. Bahn, and S. H. Noh, "A Unified Buffer Cache Architecture That Subsumes Journaling Functionality via Nonvolatile Memory," *ACM Transactions on Storage (TOS)*, vol. 10, no. 1, pp. 1:1-1:17, January 2014.

[8] R. Fang, H. Hsiao, B. He, C. Mohan, and Y. Wang, "High Performance Database Logging Using Storage Class Memory," *in Proceedings of International Conference on Data Engineering (ICDE)*, April 2011, pp. 1221-1231.

[9] Z. Zhang, L. Ju and Z. Jia, "Unified DRAM and NVM hybrid buffer cache architecture for reducing journaling overhead," *in Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2016, pp. 942-947.

[10] X. Zhang, D. Feng, Y. Hua and J. Chen, "A Cost-Efficient NVM-Based Journaling Scheme for File Systems," *in Proceedings of IEEE International Conference on Computer Design (ICCD)*, November 2017, pp. 57-64.

[11] H. Yan, H. R. Cherian, E. C. Ahn, X. Qian, and L. Duan, "iCELIA: A Full-Stack Framework for STT-MRAM-Based Deep Learning Acceleration," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 31, no. 2, pp. 408-422, February 2020.

[12] S. Mittal and J. S. Vetter, "A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 27, no. 5, pp. 1537-1550, May 2016.

[13] E. Cheshmikhani, H. Farbeh, and H. Asadi, "A System-Level Framework for Analytical and Empirical Reliability Exploration of STT-MRAM Caches," *IEEE Transactions on Reliability (TR)*, vol. 69, no. 2, pp. 594-610, June 2020.

[14] B. Wu, Y. Cheng, J. Yang, A. Todri-Sanial, and W. Zhao, "Temperature Impact Analysis and Access Reliability Enhancement for 1T1MTJ STT-RAM," *IEEE Transactions on Reliability (TR)*, vol. 65, no. 4, pp. 1755-1768, December 2016.

[15] Z. Azad, H. Farbeh, A. M. H. Monazzah, and S. G. Miremadi, "An Efficient Protection Technique for Last Level STT-RAM Caches in Multi-Core Processors," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 28, no. 6, pp. 1564-1577, June 2017.

[16] R. Salkhordeh, O. Mutlu, and H. Asadi, "An Analytical Model for Performance and Lifetime Estimation of Hybrid DRAM-NVM Main Memories," *IEEE Transactions on Computers (TC)*, vol. 68, no. 8, pp. 1114-1130, August 2019.

[17] E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an energy-efficient main memory alternative," *in Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, April 2013, pp. 256-267.

[18] J. Ren, J. Zhao, S. Khan, J. Choi, Y. Wu, and O. Mutlu, "ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems," *in Proceedings of International Symposium on Microarchitecture (MICRO)*, December 2015, pp. 672-685.

[19] Y. Kwon, H. Fingler, T. Hunt, S. Peter, E. Witchel, and T. Anderson, "Strata: A Cross Media File System," *in Proceedings of Symposium on Operating Systems Principles (SOSP)*, October 2017, pp. 460-477.

[20] Y. J. Lin, C.-L. Yang, H. P. Li, and C. Y. M. Wang, "A Hybrid DRAM/PCM Buffer Cache Architecture for Smartphones with QoS Consideration," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol.22, no. 2, pp. 27-49, March 2017.

[21] C. C. Ho, Y. M. Chang, Y. H. Chang, and H. C. Chen, and T. W. Kou, "Write-Aware Memory Management For Hybrid SLC-MLC PCM Memory Systems," ACM SIGAPP Applied Computing Review, vol.17, no. 2, pp.16-26, February 2017.

[22] S. Song, A. Das, O. Mutlu, and N. Kandasamy, "Enabling and Exploiting Partition-Level Parallelism (PALP) in Phase Change Memories," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 53:1-53:25, October 2019.

[23] H. Yoon, J. Meza, N. Muralimanohar, P. N. Jouppi, and O. Mutlu, "Efficient Data Mapping and Buffering Techniques for Multilevel Cell Phase-Change Memories," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 11, no. 4, pp. 40:1-40:25, January 2015.

[24] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory As a Scalable Dram Alternative," *in Proceedings of International Symposium on Computer Architecture (ISCA)*, June 2009, pp. 2-13.

[25] H. Farbeh, A. M. H. Monazzah, E. Aliagha, and E. Cheshmikhani, "A-CACHE: Alternating Cache Allocation to Conduct Higher Endurance in NVM-based Caches," *IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS-II)*, vol. 66, no. 7, pp. 1237-1241, July 2019.

[26] M. Bazzaz, A. Hoseinghorban, F. Poursafaei, and A. Ejlali, "High-Performance Predictable NVM-based Instruction Memory for Real-Time Embedded Systems," *IEEE Transactions on Emerging Topics in Computing (TETC)*, vol. 9, no. 1, pp. 441-455, January 2021.

[27] R. Salkhordeh and H. Asadi, "An Operating System level data migration scheme in hybrid DRAM-NVM memory architecture," *in Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2016, pp. 936-941.

[28] Intel, "Intel® Optane™ SSD 905P Series," [Online]. Available: https://ark.intel.com/content/www/un/en/ark/products/147529/intel-optane-ssd-905p-series-960gb-2-5in-pcie-x4-3d-xpoint.html. Accessed: 05/08/2020.

[29] J. Boukhobza, S. Rubini, R. Chen, and Z. Shao, "Emerging NVM: A survey on architectural integration and research challenges," *ACM Transactions on Design Au-*

tomation of Electronic Systems (TODAES), vol. 23, no. 2, pp. 1-32, November 2017.

[30] G. O. Puglia, A. F. Zorzo, C. A. F. De Rose, T. Perez, and D. Milojicic, "Non-Volatile Memory File Systems: A Survey," *IEEE Access*, vol. 7, pp. 25836-25871, February 2019.

[31] M. Tarihi, H. Asadi, A. Haghdoost, M. Arjomand, and H. Sarbazi-Azad, "A Hybrid Non-Volatile Cache Design for Solid-State Drives Using Comprehensive I/O Characterization," *IEEE Transactions on Computers (TC)*, vol. 65, no. 6, pp. 1678-1691, June 2016.

[32] J. J. Kan, C. Park, C. Ching, J. Ahn, Y. Xie, M. Pakala, and S. H. Kang, "A Study on Practically Unlimited Endurance of STT-MRAM," *IEEE Transactions on Electron Devices (TED)*, vol. 64, no. 9, pp. 3639-3646, September 2017.

[33] E. Cheshmikhani, H. Farbeh, S.G. Miremadi, and H. Asadi, "TA-LRW: A Replacement Policy for Error Rate Reduction in STT-MRAM Caches", *IEEE Transactions on Computers (TC)*, vol. 68, no. 3, pp. 455-470, March 2019.

[34] MRAM-info, "MRAM Companies," [Online]. Available: https://www.mram-info.com/companies. Accessed: 24/02/2021.

[35] RRAM-info, "RRAM Chip Makers," [Online]. Available: https://www.rram-info.com/companies/rram-chip-maker s. Accessed: 24/02/2021.

[36] Cypress, "WM72016-6," [Online]. Available: https://www.cypress.com/file/120726/download. Accessed: 24/02/2021.

[37] Fujitsu, "64K (8K×8) Bit I^2C MB85RC64," [Online]. Available: https://www.fujitsu.com/downloads/ MICRO/fme/fram/datasheet-fram-mb85rc64.pdf. Accessed: 24/02/2021.

[38] C. Sliwa, "Intel's 3D XPoint dominance will face challenge from Micron," [Online]. Available: https://searchstorage.techtarget.com/news/252492911/Intels-3D-XPoint-dominance-will-face-challenge-from-Micron. Accessed: 24/02/2021.

[39] Statista, " Quarterly market share held by NAND flash memory manufacturers worldwide from 2010 to 2020," [Online]. Available: https://www.statista.com/statistics/275886/market-share-held-by-leading-nand-flash-memory-manufacturers-worldwide. Accessed: 20/02/2021.

[40] STMicroelectronics, "STMicroelectronics Now Sampling Embedded PCM for Automotive Microcontrollers," [Online]. Available: https://www.st.com/content/st_com/en/about/media-center/press-item.dispfoldersel.html/t4119.html. Accessed: 22/02/2021.

[41] S. Sills, A. Calderoni, N. Ramaswamy, S. Yasuda, and K. Aratani, "High-density reRAM for storage class memory," *in Proceedings of Non-Volatile Memory Technology Symposium (NVMTS)*, October 2015, pp. 1-4.

[42] S. Chen, T. Chen, Y. Chang, H. Wei, and W. Shih, "A Partial Page Cache Strategy for NVRAM-Based Storage Devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 2, pp. 373-386, February 2020.

[43] H. Naeimi, C. Augustine, A. Raychowdhury, S. L. Lu, and J. Tschanz, "STT-MRAM Scaling and Retention Failure," *Intel Technology Journal (ITJ)*, vol. 17, no. 1, 2013, pp. 54-75.

[44] N. Sayed, S. M. Nair, R. Bishnoi, and M. B. Tahoori, "Process variation and temperature aware adaptive scrubbing for retention failures in STT-MRAM," *in Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC)*, January 2018, pp. 203-208.

[45] X. Guo, M. N. Bojnordi, Q. Guo, and E. Ipek, "Sanitizer: Mitigating the Impact of Expensive ECC Checks on STT-MRAM Based Main Memories," *IEEE Transactions on Computers (TC)*, vol. 67, no. 6, pp. 847-860, June 2018.

[46] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing Non-Volatility for Fast and Energy-efficient STT-RAM Caches," *in Proceedings of IEEE International Symposium on High Performance Computer Architecture (HPCA)*, February 2011, pp. 50-61.

[47] T. Lee and S. Yoo, "Selective Refresh to Avoid Read Disturb Errors in STT-RAM Main Memory," *in Proceedings of International SoC Design Conference (ISOCC)*, October 2016, pp. 315-316.

[48] H. Yan, L. Jiang, L. Duan, W. M. Lin, and E. John, "FlowPaP and FlowReR: Improving Energy Efficiency and Performance for STT-MRAM-Based Handheld Devices under Read Disturbance," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 132:1-132:20, September 2017.

[49] D. Narayanan, A. Donnelly, and A. Rowstron, "Write Off-Loading: Practical Power Management for Enterprise Storage," *ACM Transactions on Storage (TOS)*, vol. 4, no. 3, pp. 1-23, November 2008.

[50] S. Ahmadian, O. Mutlu, and H. Asadi, "ECI-Cache: A High-Endurance and Cost-Efficient I/O Caching Scheme for Virtualized Platforms," *in Proceedings of the ACM on Measurement and Analysis of Computing Systems (SIGMETRICS)*, June 2018, pp. 1-34.

[51] Y. Cai, Y. Luo, S. Ghose, and O. Mutlu, "Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery," *in Proceedings of International Conference on Dependable Systems and Networks (DSN)*, June. 2015, pp. 438-449.

[52] J. Wan, W. Wu, L. Zhan, Q. Yang, X. Qu, and C. Xie, "DEFTCache: A Cost-effective and Highly Reliable SSD Cache for RAID Storage," *in Proceedings of IEEE International Parallel Distributed Processing Symposium (IPDPS)*, May 2017, pp. 102-111.

[53] E. Cheshmikhani, A. M. Hosseini Monazah, H. Farbeh, and S. G. Miremadi, "Investigating the Effects of Process Variations and System Workloads on Reliability of STT-RAM Caches," *in Proceedings of European Dependable Computing Conference (EDCC)*, September 2016, pp. 120-129.

[54] Businesswire, "Emerging Memories Ramp Up Report, 2019-2029 - Manufacturing Equipment Revenue to Rise from an Estimated $26M in 2018 to Between $238M to $1.4B by 2029- ResearchAndMarkets.com," [Online]. Available: https://www.businesswire.com/news/home/20200123005593/en/Emerging-Memories-Ramp-Report-2019-2029—Manufacturing. Accessed: 21/03/2020.

[55] M. Hadizadeh, E. Cheshmikhani, and H. Asadi, "STAIR: High Reliable STT-MRAM Aware Multi-Level

I/O Cache Architecture by Adaptive ECC Allocation," *in Proceedings of Design, Automation & Test in Europe Conference & Exhibition* (DATE), March 2020, pp. 1484-1489.

[56] E. Cheshmikhani, H. Farbeh, and H. Asadi, "ROBIN: Incremental Oblique Interleaved ECC for Reliability Improvement in STT-MRAM Caches," *in Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC)*, January 2019, pp. 1-6.

[57] E. Cheshmikhani, H. Farbeh, and H. Asadi, "Enhancing Reliability of STT-MRAM Caches by Eliminating Read Disturbance Accumulation," *in Proceedings of Design, Automation and Test in Europe Conference (DATE)*, March 2019, pp. 854-859.

[58] Z. Liu, W. Wen, L. Jiang, Y. Jin, and G. Quan, "A statistical STT-RAM retention model for fast memory subsystem designs," *in Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC)*, January 2017, pp. 720-725.

[59] E. Aliagha, A. M. H. Monazzah, and H. Farbeh, "REACT: Read/Write Error Rate Aware Coding Technique for Emerging STT-MRAM Caches," *IEEE Transactions on Magnetics (TMAG)*, vol. 55, no. 5, pp. 1-8, May 2019.

[60] J. Li, P. Ndai, A. Goel, S. Salahuddin, and K. Roy, "Design Paradigm for Robust Spin-Torque Transfer Magnetic RAM (STT MRAM) from Circuit/Architecture Aerspective," *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems*, vol. 18, no. 12, pp. 1710-1723, December 2010.

[61] J. Axboe, "'Flexible IO Tester (fio)," [Online]. Available: https://github.com/axboe/fio. Accessed: 13/12/19.

[62] V. Tarasov, E. Zadok, and S. Shepler, "Filebench: A flexible framework for file system benchmarking," *USENIX; login*, vol. 41, no. 1, pp. 6-12, Spring 2016.

[63] J. Katcher, "Postmark: A New File System Benchmark," Technical Report TR3022, Network Appliance, vol. 8, October 1997.

[64] K. Asifuzzaman, R. S. Verdejo, and P. Radojkovic, "Enabling a reliable STT-MRAM main memory simulation," *3rd ACM International Symposium on Memory Systems (MEMSYS)*, New York, NY, USA, pp. 283-292, Oct. 2017.

[65] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator," *IEEE Computer Architecture Letters (CAL)*, vol. 10, no. 1, pp. 16- 19, March 2011.

[66] D. H. Kang, S. J. Han, Y. C. Kim and Y. I. Eom, "CLOCK-DNV: A Write Buffer Algorithm for Flash Storage Devices of Consumer Electronics," *IEEE Transactions on Consumer Electronics (TCE)*, vol. 63, no. 1, pp. 85-91, February 2017.

[67] S. Park, D. Jung, J. Kang, J. Kim, and J. Lee, "CFLRU: A Replacement Algorithm for Flash Memory," *in Proceedings of International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, October 2006, pp. 234-241.

[68] A. Chatzistergiou, M. Cintra, and S. D. Viglas, "REWIND: Recovery Write-Ahead System for In-Memory Non-Volatile Data-Structures," *in Proceedings of the VLDB Endowment*, vol. 8, no. 5, January 2015.

[69] J. Xu and S. Swanson, "NOVA: A Log-structured File System for Hybrid Volatile/Non-volatile Main Memories," *in Proceedings of USENIX Conference on File and Storage Technologies (FAST)*, February 2016, pp. 323-338.

[70] Z. Zhang, Z. Shen, Z. Jia, and Z. Shao, "UniBuffer: Optimizing Journaling Overhead With Unified DRAM and NVM Hybrid Buffer Cache," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 9, pp. 1792-1805, September 2020.

[71] S. Chen, Y. Chang, Y. Chang, and W. Shih, "mwJFS: A Multiwrite-Mode Journaling File System for MLC NVRAM Storages," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 9, pp. 2060-2073, September 2019.

[72] A. Jog, A. K. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer, and C. R. Das, "Cache Revive: Architecting Volatile STT-RAM Caches for Enhanced Performance in CMPs," *in Proceedings of Design Automation Conference (DAC)*, June 2012, pp. 243-252.

[73] H. Asadi, E. Cheshmikhani, and H. Farbeh, "Preventing Read Disturbance Accumulation in a Cache Memory," US Patent, App. No. 16798451, June 2020.

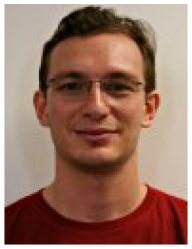
[74] E. Cheshmikhani, H. Farbeh, and H. Asadi, "3RSeT: Read Disturbance Rate Reduction in STT-MRAM Caches by Selective Tag Comparison," *IEEE Transactions on Computers (TC)*, In Press, 2021.

[75] Z. Chen, Y. Lu, and F. Liu, "Me-CLOCK: A Memory-Efficient Framework to Implement Replacement Policies for Large Caches," *IEEE Transactions on Computres (TC)*, vol. 65, no. 8, pp. 2665-2671, Aug. 2017.

Mostafa Hadizadeh received the B.Sc. degree in computer engineering from Shahid Beheshti University (SBU), Tehran, Iran, in 2016, and the M.Sc. degree in computer engineering at Sharif University of Technology (SUT), Tehran, Iran, in 2018. He is a member of Data Storage, Networks, and Processing (DSN) Laboratory since 2017. From December 2016 to May 2017, he was a member of Dependable Systems Laboratory (DSL) at SUT. His research interests include computer architecture, data storage & memory systems, systems on chip, and dependable system design.

Elham Cheshmikhani received the B.Sc. degree in computer engineering from Iran University of Science and Technology (IUST), the M.Sc. degree in computer engineering from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2011 and 2013, respectively and the PhD degree in computer engineering from Sharif University of Technology (SUT), Tehran, Iran in Feb. 2020. She was a member of the Design and Analysis of Dependable Systems (DADS) at AUT from 2011 to 2015 and has been a member of the Dependable Systems Laboratory (DSL) and Data Storage, Networks & Processing Laboratory (DSN) since 2015 and 2017, respectively. Her research interests include emerging nonvolatile memory technologies, dependability analysis, fault tolerance, and storage systems. More recently, she received the Best Paper Award at IEEE/ACM Design, Automation, and Test in Europe (DATE) in 2019.

Maysam Rahmanpour received a B.Sc. degree in computer engineering from Shahid Beheshti University (SBU), Tehran, Iran in 2016. He received an M.Sc. degree in computer engineering from Sharif University of Technology (SUT), Tehran, Iran. He is a member of the Data Storage, Networks, and Processing (DSN) Laboratory since December 2016. His research interest includes Computer Architecture, Emerging NVM-Based Architecture, Memory Systems, and High-Performance Systems.



Onur Mutlu is a Professor of Computer Science at ETH Zurich. He is also a faculty member at Carnegie Mellon University, where he previously held the Strecker Early Career Professorship. His current broader research interests are in computer architecture, systems, hardware security, and bioinformatics. A variety of techniques he, along with his group and collaborators, has invented over the years have influenced industry and have been employed in commercial microprocessors and memory/storage systems. He obtained his PhD and MS in ECE from the University of Texas at Austin and BS degrees in Computer Engineering and Psychology from the University of Michigan, Ann Arbor. He started the Computer Architecture Group at Microsoft Research (2006-2009), and held various product and research positions at Intel Corporation, Advanced Micro Devices, VMware, and Google. He received the inaugural IEEE Computer Society Young Computer Architect Award, the inaugural Intel Early Career Faculty Award, US National Science Foundation CAREER Award, Carnegie Mellon University Ladd Research Award, faculty partnership awards from various companies, and a healthy number of best paper or "Top Pick" paper recognitions at various computer systems, architecture, and hardware security venues. He is an ACM Fellow "for contributions to computer architecture research, especially in memory systems", IEEE Fellow for "contributions to computer architecture research and practice", and an elected member of the Academy of Europe (Academia Europaea). His computer architecture and digital circuit design course lectures and materials are freely available on YouTube, and his research group makes a wide variety of software and hardware artifacts freely available online. For more information, please see his webpage at https://people.inf.ethz.ch/omutlu/.

Hossein Asadi (M'08, SM'14) received the BSc and MSc degrees in computer engineering from the SUT, Tehran, Iran, in 2000 and 2002, respectively, and the PhD degree in computer engineering from Northeastern University, Boston, MA, USA, in 2007.

He was with EMC Corporation, Hopkinton, MA, as a research scientist and senior hardware engineer, from 2006 to 2009. From 2002 to 2003, he was a member of the Dependable Systems Laboratory, SUT, where he researched hardware verification techniques. From 2001 to 2002, he was a member of the Sharif Rescue Robots Group. He has been with the Department of Computer Engineering, SUT, since 2009, where he is currently a full professor. He is the founder and director of the *Data Storage, Networks, and Processing* (DSN) Laboratory and the director of Sharif *High-Performance Computing* (HPC) Center. He spent three months in the summer 2015 as a Visiting Professor at the School of Computer and Communication Sciences at EPFL. He is also the co-founder of HPDS corp., designing and fabricating midrange and high-end data storage systems. He has authored and co-authored more than eighty technical papers in reputed journals and conference proceedings and holds several international patents. His current research interests include data storage systems and networks, solid-state drives, operating system support for I/O and memory management, and high-performance, reconfigurable, and dependable computing.

Dr. Asadi was a recipient of the Technical Award for the Best Robot Design from the International RoboCup Rescue Competition, organized by AAAI and RoboCup, a recipient of Best Paper Award at the 15th CSI International Symposium on *Computer Architecture & Digital Systems* (CADS), the Distinguished Lecturer Award from SUT in 2010, the Distinguished Researcher Award and the Distinguished Research Institute Award from SUT in 2016, the Distinguished Technology Award from SUT in 2017, and the Distinguished Research Lab Award from SUT in 2019. He is also recipient of Extraordinary Ability in Science visa from US Citizenship and Immigration Services in 2008. He has been ranked among "Top-10" among 500+ faculties by Research and Technology Deputy, Sharif University of Technology for five consecutive years from 2016 to 2020. More recently, he received the Best Paper Award at IEEE/ACM Design, Automation, and Test in Europe (DATE) in 2019. He has served as a guest editor of IEEE Transactions on Computers, an Associate Editor of Microelectronics Reliability, a Program Co-Chair of CADS2015, and the Program Chair of CSI National Computer Conference (CSICC2017). He is a senior member of the IEEE.