TA-LRW: A Replacement Policy for Error Rate Reduction in STT-MRAM Caches

Elham Cheshmikhani, Hamed Farbeh, Seyed Ghassem Miremadi, *Senior Member, IEEE,*
Hossein Asadi, *Senior Member, IEEE*

Abstract—As technology process node scales down, on-chip SRAM caches lose their efficiency because of their low scalability, high leakage power, and increasing rate of soft errors. Among emerging memory technologies, $Spin\text{-}Transfer\ Torque\ Magnetic\ RAM$ (STT-MRAM) is known as the most promising replacement for SRAM-based cache memories. The main advantages of STT-MRAM are its non-volatility, near-zero leakage power, higher density, soft-error immunity, and higher scalability. Despite these advantages, high error rate in STT-MRAM cells due to $retention\ failure$, $write\ failure$, and $read\ disturbance$ threatens the reliability of cache memories built upon STT-MRAM technology. The error rate is significantly increased in higher temperature, which further affects the reliability of STT-MRAM-based cache memories. The major source of heat generation and temperature increase in STT-MRAM cache memories is write operations, which are managed by cache $replacement\ policy$. To the best of our knowledge, none of previous studies have attempted to mitigate heat generation and high temperature of STT-MRAM cache memories using replacement policy. In this paper, we first analyze the cache behavior in conventional $Least\text{-}Recently\ Used$ (LRU) replacement policy and demonstrate that the majority of consecutive write operations (more than 66%) are committed to adjacent cache blocks. These adjacent write operations cause accumulated heat and increased temperature, which significantly increase the cache error rate. To eliminate heat accumulation and the adjacency of consecutive writes, we propose a cache replacement policy, named $Thermal\text{-}Aware\ Least\text{-}Recently\ Written$ (TA-LRW), to smoothly distribute the generated heat by conducting consecutive write operations in distant cache blocks. TA-LRW guarantees the distance of at least three blocks for each two consecutive write operations in an 8-way associative cache. This distant write scheme reduces the temperature-induced error rate by 94.8%, on average, compared with the conventional LRU policy, which results in 6.9x reduction in cache error rate. The implementation cost and complexity of TA-LRW is as low as $First\text{-}In,\ First\text{-}Out$ (FIFO) policy while providing a near-LRU performance, having the advantages of both replacement policies. The significantly reduced error rate is achieved by imposing only 2.3% performance overhead compared with the LRU policy.

Index Terms—Cache memory, heat accumulation, read disturbance, replacement policy, retention failure, STT-MRAM, write failure.

1 Introduction

Cache memories occupy a large part of the processor chip area and play an important role in performance, reliability, and power consumption of the whole computer systems [1], [2]. Conventional SRAM caches face several challenges by downscaling technology process node, e.g., high leakage power, increased error rate, low density, and high sensitivity to process variations. Recent developments have suggested emerging memory technologies, e.g., $Phase\ Change\ Memory$ (PCM), $Resistive\ RAM$ (ReRAM), $Ferroelectric\ RAM$ (FeRAM), and $Spin\text{-}Transfer\ Torque\ Random\ Access\ Memories$ (STT-MRAM), as replacements for currently used memories [3]–[5]. Among these memories, STT-MRAM is known as the most promising alternative for SRAMs in on-chip caches according to recent reports [6]–[8]. Several recent studies have evaluated STT-MRAM and SRAM as *Last-Level Caches* (LLCs) and illustrated that STT-MRAM outperforms SRAM in terms of energy consumption and performance. STT-MRAM improves the performance by increasing the cache capacity (due to its higher density) and reduces the total energy consumption due to its near-zero leakage power [9]–[14].

Unlike SRAM technology, STT-MRAM cells are non-volatile and their leakage power is negligible. In addition, their density is higher than that of SRAMs, and they are not vulnerable to radiation-induced soft errors. However, the high error rate of STT-MRAM cells should be addressed to make it applicable in on-chip caches. STT-MRAM caches are error-prone to three types of failures in write and read accesses as well as memory idle intervals as follows: 1) Due to stochastic behavior of magnetization process of STT-MRAM cells, a cache cell may not switch on a write operation, resulting to a $write\ failure$ [15], [16]; 2) A cache cell may unintentionally flip by the current applied for reading a cache block, which leads to a $read\ disturbance\ error$ [17], [18]; 3) Without applying any current, an idle cache cell may flip stochastically, leading to a $retention\ failure$ [19], [20]. The rates of these three types of errors are significantly increased by raising the temperature. Increasing the temperature reduces the $thermal\ stability\ factor$ (Δ) of STT-MRAM cells, which results in exponentially increase in read disturbance

- *E. Cheshmikhani and H. Asadi are with the Department of Computer Engineering, Sharif University of Technology, Tehran 11155-11365, Iran. E-mail: echeshmikhani@ce.sharif.edu; asadi@sharif.edu*
- *H. Farbeh is with the Department of Computer Engineering, Amirkabir University of Technology, Tehran 15916-34311, Iran. E-mail: farbeh@aut.ac.ir*
- *S. G. Miremadi was with the Department of Computer Engineering, Sharif University of Technology, Tehran 11155-11365, Iran (deceased on April 13, 2017). E-mail: miremadi@sharif.edu*

Manuscript received December 27, 2017; revised June 13, 2018; revised July 29, 2018.

and retention failure rates. On the other hand, write current decreases in higher temperature, which results to a higher write failure rate [21].

The major source of heat generation and high temperature in STT-MRAM caches is read and write accesses [22]–[26].The energy dissipated per write access in the cache is by one order of magnitude higher than that per read access. Therefore, the increase in cache temperature is mainly due to write operations. The write operations affect the temperature and consequently the cache error rate in two aspects: 1) the number of write requests to the cache per unit of time for an arbitrary cache configuration and 2) the distribution pattern of the writes over the cache blocks. Although the former is beyond the control of the cache, the latter is handled by the cache controller. One of the main factors in write distribution non-uniformity is the decisions made by cache replacement policy, as it determines into which cache block the incoming data should be written. To the best of our knowledge, none of the existing cache replacement policies are aware of the temperature and the heat generated by write operations. Hence, it is highly probable that consecutive write accesses are performed in adjacent cache blocks, which leads to heat accumulation in some cache regions and the increased error rate.

In this paper, we first conduct a set of experiments using finite-volume based $Computational\ Fluid\ Dynamics$ (CFD) tools (ANSYS FLUENT 14.5 and GAMBIT FLUENT [27]) and show that each write operation into a cache block increases the temperature of the written STT-MRAM cells by 9°K. Then, we investigate the sequence of L2 cache write operations in conventional $Least\text{-}Recently\ Used$ (LRU) replacement policy and demonstrate that a large fraction of consecutive incoming data are written into the same or adjacent cache blocks. These write operations, originated by a cache miss or writeback from L1 caches, accumulate the generated heat locally and increase the error rate in hot blocks. Using these observations, we propose a cache replacement policy, so-called $Thermal\text{-}Aware\ Least\text{-}Recently\ Written$ (TA-LRW), to uniformly distribute the temperature and prevent the heat accumulation. Unlike LRU policy, TA-LRW policy sorts the blocks based on their last write access instead of their last read/write access and evicts the least-recently written block on a cache miss. On a writeback hit, the incoming data is written in the least-recently written block instead of overwriting the block containing the previous version of data. To prevent the heat accumulation, the target block to eviction is selected *physically far enough*[1] from the previously written block in the cache set.

TA-LRW policy uses a pointer to indicate the next block to be written in a set. To prevent the heat accumulation, we present a permutation of write sequence in which the distance of two consecutive writes is at least three blocks and the pointer is updated after each write based on this permutation. TA-LRW policy does not require the complicated LRU policy controller and peripherals and its implementation is as simple as the low-cost $First\text{-}In,\ First\text{-}Out$ (FIFO) policy. Meanwhile, the performance of TA-LRW is very close to that of LRU and significantly higher than that of FIFO, since it decides based on write access history, which we show that is an effective approximation for the total access history.

We evaluate the proposed TA-LRW policy by running a set of workloads from SPEC CPU2006 benchmark suite [28] using gem5 cycle-accurate simulator [29]. The STT-MRAM cache parameters are extracted from NVSim tool [30] and the cache temperature is measured using ANSYS FLUENT 14.5 and GAMBIT FLUENT tools [27]. The L2 cache configuration is adjusted according to STT-MRAM parameters and the cache is modified to operate based on TA-LRW policy. The simulation results show that the TA-LRW policy reduces the temperature-induced error rate by 6.9x compared with LRU policy. Furthermore, TA-LRW increases the miss rate and $Cycles\ Per\ Instruction$ (CPI) by only 0.5% and 2.3%, respectively, compared with LRU, while these values for FIFO are 9.5% and 10.3%, respectively.

Briefly, the **main contributions** of this work are as follows:

1) This is the first study that investigates the effect of temperature on STT-MRAM cache error rate and demonstrates that heat accumulation increases the error rate by 110.9%. We also illustrate that this heat accumulation is mainly due to locality of committed write operations in the cache.
2) As a first effort, this work models the STT-MRAM cells using CFD tools to analyze its temperature behavior. Using these finite-volume based tools, we show that each write operation increases the cell temperature by 9°K. This significant increase in STT-MRAM cell temperature implies that write operations are the main source of heat generation in STT-MRAM-based caches.
3) We reveal that using conventional LRU replacement policy, write operations cause a substantial increase in error rate due to heat accumulation in L2 cache. Our evaluations show that more than 66% of write operations are performed in the hottest block or in its adjacent blocks in LRU replacement policy.
4) We propose a simple yet effective replacement policy, called TA-LRW, to prevent the heat accumulation in the cache. Unlike conventional replacement policies, TA-LRW decides to write into the blocks that are far enough apart to minimize the accumulated heat and reduce the cache error rate. We explore all possible permutations for write sequences and select the most suitable permutation that guarantees the minimum heat accumulation in TA-LRW policy.
5) Detailed evaluations and comparisons are conducted to show the efficiency of TA-LRW. Our evaluations show that the conventional LRU replacement policy increases the STT-MRAM cache error rate by 110.9%, which is reduced to 16.1% by TA-LRW. These values indicate 6.9x reduction in temperature-induced error rate.

The rest of this paper is organized as follows. Section 2 describes the basics of STT-MRAM and its reliability challenges. In Section 3, the observations and motivations for this work are discussed. The details of the proposed TA-LRW policy are presented in Section 4. Section 5 gives the simulation setup and evaluation results. Related work and

1. Two blocks are physically far enough if the effect of generated heat during the write operation in one of them is negligible on the temperature of the other block.

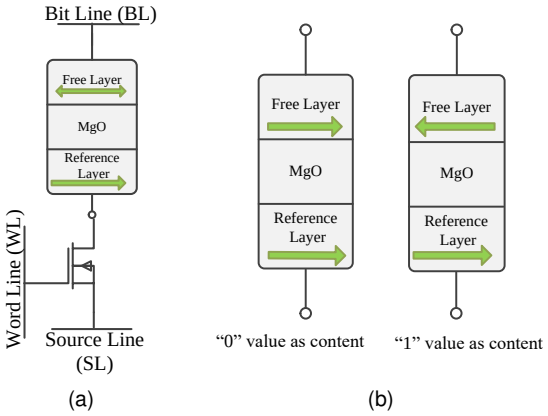


Fig. 1. STT-MRAM schematic [4]: (a) 1T1J STT-MRAM structure and (b) MTJ states.

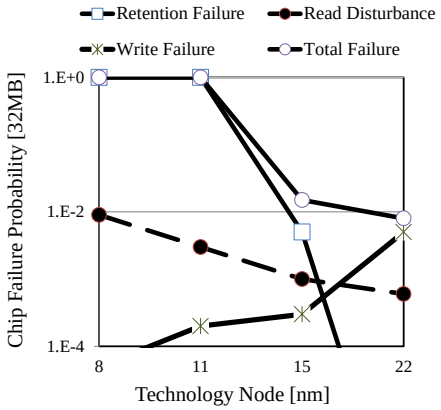


Fig. 2. STT-MRAM failure rates in technology node scaling [39].

discussions are investigated in Section 6. Finally, we conclude the paper in Section 7.

2 Preliminaries

2.1 STT-MRAM Basics

Magnetoresistive RAM (MRAM) cells store data bits using magnetic charge instead of electrical charge. These magnetic storage elements, known as $Magnetic\ Tunnel\ Junction$ (MTJ), store the charge and are formed from two ferromagnetic layers, separated by a thin oxide barrier layer [31], [32]. The barrier layer is made up of crystallized MgO. There are some methods to torque the MRAM cell. The preferred technique is $Spin\text{-}Transfer\ Torque$ (STT) that uses spin-aligned electrons to torque the domain [33]. STT-MRAM cell is comprised of a MTJ and an access transistor, which is named 1T1J STT-MRAM, as shown in Fig. 1(a). One of the ferromagnetic layers in MTJ, named $reference\ layer$, has a fixed magnetic field direction and the field direction of the other layer, named $free\ layer$, can be changed through an applied current [34], [35].

The resistance of the MTJ alternates between a low and high values based on the magnetic field direction of the free layer. If it is parallel with the magnetic field direction of the fixed layer, the MTJ is in low resistance state and if is antiparallel with that, the resistance of MTJ is high. The two states of the MTJ unit are shown in Fig. 1(b). These high and low resistances are interpreted as binary logic '1' and '0', respectively. High resistance stores logic '1' in the MTJ and low resistance is assumed as logic '0', as shown in Fig. 1(b).

2.2 Read and Write Operations

The resistance of the MTJ shows the cell value. To read the data stored in a cell, word line is set to V_{DD} to turn on the access transistor. Then, a small read current/voltage is applied to the bit line [4], [36]. This current/voltage generates a current in the cell, which can be compared to a reference value. If the sensed voltage/current is higher than the reference, the MTJ resistance is high and the cell contains '1'. Otherwise, the resistance is low and the cell contains '0'.

To write a value in a STT-MRAM cell, the magnetic field direction of the free layer should be changed. To this end, a write voltage/current is applied to the bit line or source line. If the current flows from source line to bit line (voltage is applied to source line), electron charges flow from the free layer to the reference layer. Due to the strong magnetic field in reference layer, the electrons with the opposite direction of the reference layer are reflected to the free layer. This reflection creates a torque in the free layer, antiparallelizes the magnetic of MTJ, and leads to write '1' [37]. To write '0' in the cell, the spin-polarized current flows from bit line to source line and causes the electron charges to flow from the reference layer to the free layer. Electrons with the spin direction same as that of electrons spin in the reference layer pass through the free layer and generate a torque that parallelize the two MTJ ferromagnetic layers and leads to write '0' [37], [38].

2.3 Reliability of STT-MRAM

The major reliability challenges in STT-MRAM cells are $retention\ failure$, $read\ disturbance$, and $write\ failure$ [4], [20], [39]. A retention failure occurs when the content of an idle cell unintentionally flips without applying any voltage/current. A read disturbance occurs when the content of a cell is flipped during a read operation. Due to the stochastic switching behavior of a cell, it is probable that its content flips even by a small read current. The error remains until the next write operation in the cell [20], [39]. Write operation in a STT-MRAM cell is also stochastic. Write failure is inability of a cell to switch when the write current is applied [40]. This means that the value of the cell is not changed during the write pulse interval. In the subsequent requests to the cell, an erroneous value is read until the next write operation.

Fig. 2 shows a prediction for the rates of three mentioned errors in different technology process nodes for a 32MB STT-MRAM cache memory [39]. As depicted, write failure and read disturbance are the main reliability threat in 22nm technology process node, while the rate of retention failure is negligible. However, the retention failure rate increases significantly by technology downscaling and is higher than both other two errors in 15nm technology process node and beyond. On the other hand, write failure rate decreases while read disturbance rate increases in smaller technology process nodes. The total error rate considering all three sources of errors is significantly increased by technology downscaling, in which the retention failure is the dominant contributor.

3 MOTIVATION

Retention failure occurs stochastically and increases in higher temperature. Thermal effects cause a flip in a STT-MRAM cell that follows Poisson distribution with timing characteristics of τe^{Δ} [39]. Equation (1) shows the probability of n times bit flip in the unit of time t.

$$P_{bit-flip} = \frac{\lambda^n e^{-\lambda}}{n!} \quad (1)$$

where $\lambda = t/(\tau e^{\Delta})$ is failure rate and τ is attempt period, which is assumed to be 1ns [39]. If the number of probable bit-flips goes to infinity and it occurs for odd numbers to cause an error, the retention failure probability during time t is calculated according to (2) [39].

$$P_{Ret-Failure} = \sum_{k=0,n=2k+1}^{\infty} P_{flip}(n) = 1 - exp(\frac{-t}{e^{\Delta}}) \quad (2)$$

The approximated retention failure probability equation shows that this failure rate is exponentially dependent on the thermal scalability factor (Δ). Thermal stability factor is according to (3) [39].

$$\Delta = \frac{E_b}{KT} \quad (3)$$

where, E_b is barrier energy, K is Boltzman constant, and T is temperature. Thermal stability factor (Δ) is inversely proportional to temperature and reduces by increasing the temperature. As observed in (2), reduction in Δ exponentially increases the retention failure rate. Fig. 3 depicts the $Mean\ Time\ To\ Failure$ (MTTF) of a STT-MRAM cell for some known values of Δ and shows how MTTF is reduced in higher temperature. For example, MTTF is about 10 years in 300°K for a STT-MRAM with nominal Δ of 40 and is reduced to about two days when the temperature increases to 360°K.

Next, we explore the temperature effects on read disturbance and write failure rate. The occurrence probability of a read disturbance in a STT-MRAM cell is according to (4) [41].

$$P_{Read-Disturbance} = 1 - exp(\frac{-t_{read}}{\tau} \times exp(\frac{-\Delta(I_{read} - I_{C_0})}{I_{C_0}})) \quad (4)$$

where, τ is attempt period and assumed to be 1ns, I_{read} is read current, I_{C0} is the current needed to write in 0°K and t_{read} is the read pulse. As depicted, read disturbance rate exponentially increases by Δ reduction; and, Δ reduces by increasing the temperature, as mentioned earlier. The occurrence probability of a write failure for a STT-MRAM cell is shown in (5) [42], [43].

$$P_{Write-Failure} = exp(-t_{write} \times \frac{2 \times \mu_\beta \times p \times (I_{write} - I_{C_0})}{c + \log_e(\pi^2 \times \Delta/4) \times (e \times m \times (1 + p^2))}) \quad (5)$$

where, I_{write} is write current, c is Euler constant, e is electron charge, m is magnetic momentum of the free layer, p is tunneling spin polarization, μ_β is Bohr magneton, and t_{write} is write pulse width.

Unlike retention failure and read disturbance, reducing Δ decreases the rate of write failure. However, by increasing the temperature, which results in Δ reduction, I_{write} is also reduced due to drivability degradation of the access transistor. As demonstrated in [21], the adverse effect of temperature on I_{write} is larger than its positive effect on Δ and the write failure rate increases in higher temperature. As discussed, increasing in temperature, which is mainly due to the heat generated by write operations, significantly increases the total error rate in STT-MRAM cells.

To quantify the effects of write operations on the cache temperature, we conduct a set of experiments using *Computational Fluid Dynamics* (CFD) tools and calculate the temperature of a STT-MRAM cell for a write operation. Simulation parameters which are obtained from NVSim simulator [30] are depicted in Table 1. According to Fig. 4, simulation results show that the STT-MRAM cell temperature increases from about 318.5°K to 327.5°K per write operation. This 9°K increase in the cell temperature can significantly increase the error rates in the cache blocks.

Although a write operation increases the temperature of the cache for a short interval and the block can cool down, the locality in write operations not only expands the high-temperature region, but also slows down the cooling process of hot blocks. Write operations in L2 caches are not well distributed. It is highly probable that temporally adjacent requests are written in neighboring cache blocks. The locality

TABLE 1
Details of STT-MRAM cell configuration in CFD tools simulation

Cell Area $(nm)^2$	Write Dynamic Energy (nj/access)	Write Pulse (ns)	Read Dynamic Energy (nj/access)
310.98	0.377	10	0.013

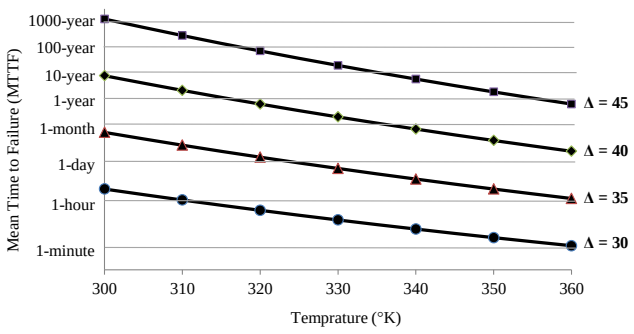


Fig. 3. Mean time to retention failure versus temperature for four different values of thermal stability factor (Δ).

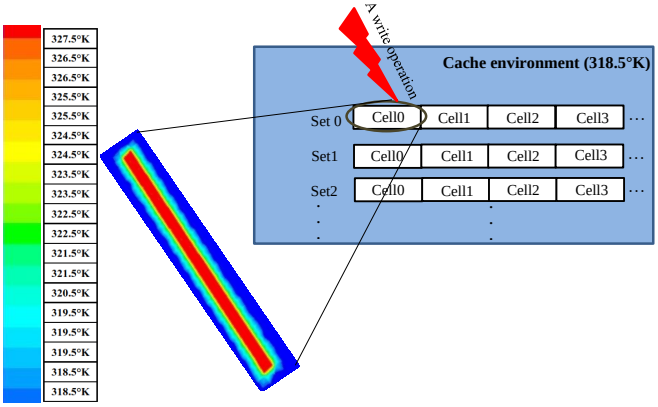
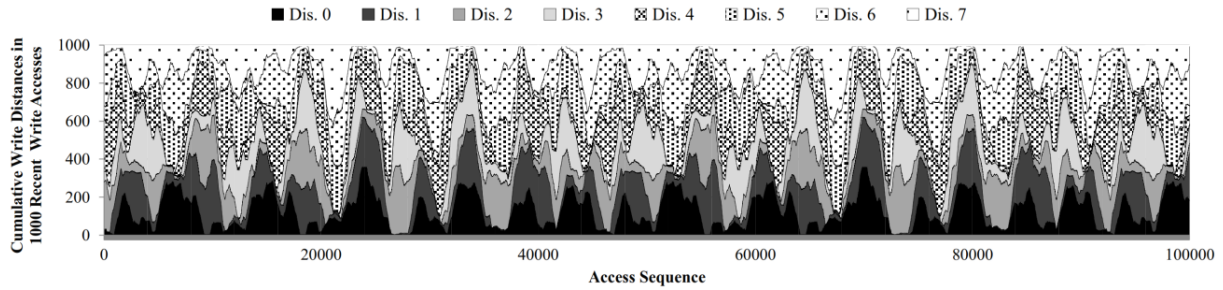
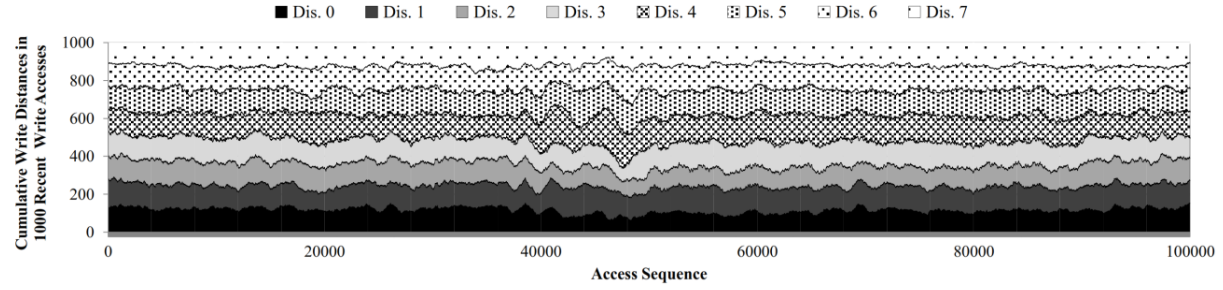


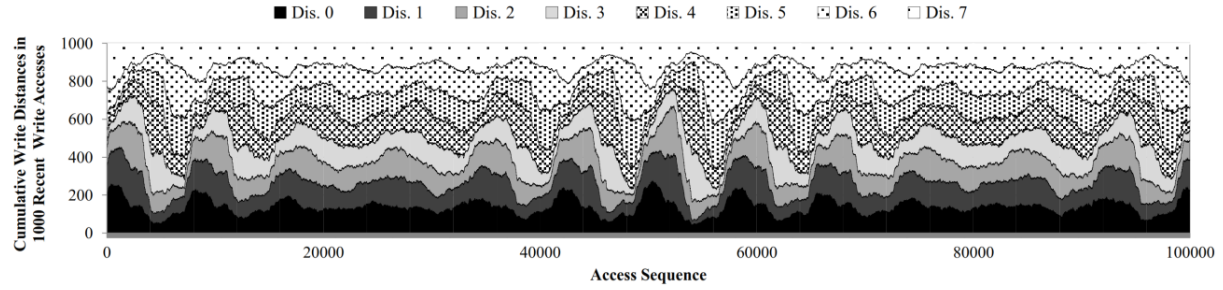
Fig. 4. Temperature increase of a cell due to a write operation.



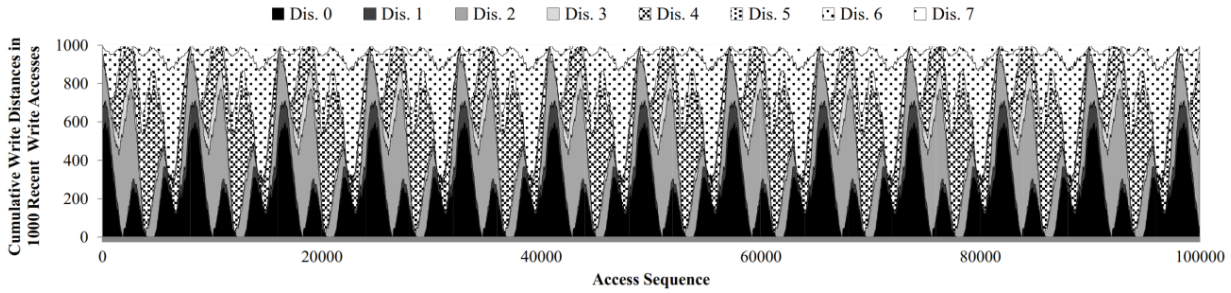
(a) bzip2



(b) gcc



(c) cactusADM



(d) lbm

Fig. 5. Accumulated write distances for every 1000 recent writes in (a) bzip2, (b) gcc, (c) cactusADM, and (d) lbm workloads in LRU policy.

in write operations leads to heat accumulation in some cache regions and increases the error rates for long intervals.

To explore non-uniformity in a sequence of write operations in L2 cache, we extract the write locations in the cache sets using gem5 cycle-accurate simulator [29] running the workloads of SPEC CPU2006 benchmark suite [28]. The details of system configuration are given in Table 2 in Section 5. Considering an 8-way set-associative cache, the distance of two consecutive writes can be any value between zero and seven. For example, if the most recent data is written in block 2 and the current data is written in block 7, the distance of these two consecutive writes is five. Fig. 5 shows the breakdown of write distances for four workloads, i.e., *bzip2*, *gcc*, *cactusADM*, and *lbm* as examples of write locality[2]. For the sake of visibility, the distances of most recent 1000 write operations is depicted in each access for the workloads and the Y-axis is the cumulative write distances for these recent writes.

Considering *bzip* workload in Fig. 5(a), there are several intervals in which the write distance of more than 20% of accesses is zero, indicating that data is written into the

2. The results for the other benchmarks are given in Fig. 16 in the Appendix.

hottest block. In addition, the zero and one write distances contribute in more than 40% of writes for considerable intervals. According to Fig. 5(b), more than 20% of incoming blocks are written into the most-recently written block (write distance 0) or into its adjacent block (write distance 1) throughout the *gcc* workload execution interval. Fig. 5(c) shows that for *cactusADM* workload, in majority of time intervals, write distances for more than 30% of writes are zero or one. Finally, there are several intervals in which more than 60% of incoming blocks are written into the hottest block in *lbm* workloads, as shown in Fig. 5(d).

These observations confirm the existence of a high temporal locality in writing into adjacent cache blocks. The above observations also indicate a high spatial locality in consecutive write operations in which a considerable number of incoming data is written into the recently written block or one of its adjacent blocks. These localities in write operations can cause heat accumulation in recently written cache regions, which in turn can increase the error rate.

4 Proposed TA-LRW Policy

Write operations in an L2 cache are from two sources, i.e., writebacks from L1 caches and read misses. On a read miss, the cache replacement policy selects a victim block to replace its content with the new incoming data. On a writeback, if the older version of the data is already available in the cache, the block will be overwritten; otherwise, the cache replacement policy determines a block that the incoming data block should be written into.

In any case, the location of the write operation in a cache set either is determined by the cache replacement policy, in the case of a cache miss, or is predetermined, in the case of a writeback hit. A large fraction of accumulated heat in cache sets is due to decisions of cache replacement policy on cache misses and the inevitable decision of cache controller for overwriting the cache blocks on writeback hits.

The goal of TA-LRW is evenly distributing the heat generated by consecutive write operations in the blocks. SLC- or MLC-based cells as well as in-plane or perpendicular cell structure do not affect the functionality of TA-LRW, since it operates on blocks of a cache set. Furthermore, as the write operation in MLC STT-MRAM caches requires larger write current and consumes more energy [44], it imposes more heat accumulation. Therefore, MLC structure in STT-MRAM memories further exacerbates the heat accumulation challenge. In addition, MLC cell faces $write\ disturbance$ as another error type besides the three well-known errors in SLC STT-MRAM, which introduces a new challenge to the STT-MRAM reliability [44], [45].

Meanwhile, it is predicted that 3D stacked design is the future structure of STT-MRAM caches [22]. Temperature-induced error rate is more severe for 3D STT-MRAM caches than its 2D counterpart due to the more difficult heat transferring in 3D STT-MRAMs. Hence, employing TA-LRW in 3D and/or MLC STT-MRAM caches is even more crucial than in their 2D and/or SLC counterparts.

4.1 Basics of TA-LRW Policy

We have already observed a high probability of heat accumulation in LRU policy, as the most conventional and widely-used replacement policy, in Fig. 5. To reduce the heat accumulation in cache sets, our approach is to write the consecutive incoming data in non-adjacent blocks. This approach requires a different policy to select victim blocks than that of the existing replacement polices as well as redirecting a writeback hit from an already available block to another block.

The primary goal of the replacement policies is to minimize the cache miss rate based on localities in accessing cache blocks. Replacing the cache blocks based on the location of the recent writes should not violate the locality principle in the cache. To this aim, our proposed cache replacement policy not only selects a suitable block to be written on a cache miss, but also redirects the overwrite operation of already available data to a suitable block. Meanwhile, as the primary goal of the existing cache replacement policies, it provides a low miss rate and high performance.

Our proposed cache replacement policy, named $Thermal\text{-}Aware\ Least\text{-}Recently\ Written$ (TA-LRW), offers four main features: 1) all consecutive incoming data are written in non-adjacent blocks, 2) the write operations are evenly distributed over all blocks in a set, 3) it replaces the cache blocks based on a semi-LRU decision to provide a near-LRU performance, 4) its implementation cost is as low as FIFO replacement policy and does not require the LRU age bits per block to keep track of access history as well as the complicated LRU controller and operations for each access. In the following, we explain the TA-LRW replacement policy in more details.

In TA-LRW policy, there is a pointer, named $write\ pointer$, in each cache set to indicate the next block that should be written. This pointer traverses over all blocks in a set band returns back to its original location. This round-robin writing strategy guarantees that no recent N incoming data blocks are written in the same cache block. To guarantee that no sequence of incoming data is written in adjacent blocks, TA-LRW updates the pointer after each write in a way that the next write is performed in a far enough block.

4.2 Write Access Manipulation

As we will show later in Section 5, the majority of writes in LRU policy are performed in the same block as that of the previous write or in their adjacent blocks. Using TA-LRW policy, it is guaranteed that the distance between two consecutive writes in a set is at least three blocks and the written blocks have enough time to cool down until their next write. To this aim, we need to find a permutation of sequence of writes into blocks in which physically adjacent blocks are located in distant places of permutation.

A set of 8-way set-associative cache is shown in Fig. 6(a). Three permutations that satisfy the TA-LRW requirements are shown in Fig. 6(b)-(d). In these permutations, the distance between each two consecutive writes is at least three blocks. For the permutation in Fig. 6(b), after one round in which all blocks are written once, the distance of four out of eight consecutive writes is three blocks and the distance of three and one out of eight consecutive writes is four and six blocks, respectively. In each round for the permutation in Fig. 6(c), the distance of five, two, and one out of eight writes is three, four, and seven blocks, respectively.

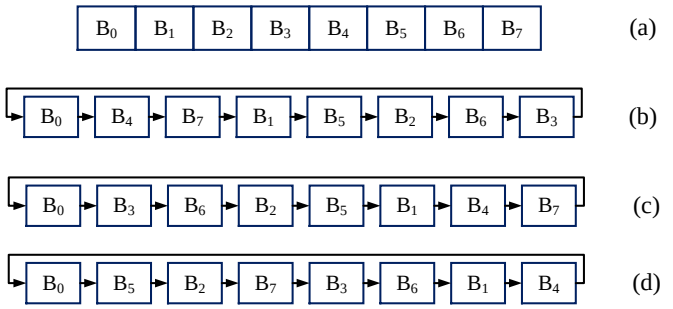


Fig. 6. Three samples of suitable permutations for TA-LRW policy: (a) cache blocks in a set, (b) first permutation for sequence of writes, (c) second permutation, (d) third permutation used for this work.

The third permutation (Fig. 6(d)) includes three different distances for the sequence of writes. The distance of three and two out of eight writes in each round is three and four, respectively, and the distance of the other three writes is five. There are several other permutations in which the write distances are three or more blocks. It also can theoretically be proven that there is no permutation with write distance of at least four. Therefore, the best permutations are those with minimum write distance of three.

We have generated all permutations (8! = 40,320) for 8-way cache out of which the minimum distance between every two adjacent number is three for 176 permutations. All of these 176 permutations guarantee that every two consecutive incoming data in a cache set are written in blocks with the distance of at least three. The implementation complexity and performance of these permutations are the same and their heat distributions are not much different. To find the best permutation between the existing 176 suitable permutations, we first analyze how a write operation in each block increases the temperature of blocks in a set based on their distance from the write location. Then, we calculate the total heat accumulation after one round of writing into all blocks for these permutations. The best choice is a permutation with minimum heat accumulation. We consider the permutation in Fig. 6(d) in this work based on the assumed values of heat accumulation for each write distance.

4.3 Architectural Aspects of TA-LRW

TA-LRW policy selects the victim block to be replaced according to its write pointer value for a read miss or writeback miss. The block indicated by the pointer is the least-recently written block in the set and is the most suitable block to be written for preventing the heat accumulation. For a writeback hit, existing replacement policies, e.g., LRU policy, have no choice other than overwriting the found block. However, it is probable that this block is among the most-recently written blocks and the current write operation leads to the accumulation of heat in the block region. TA-LRW policy overwrites the found block only if the block is selected by the write pointer. Otherwise, the block is invalidated and the incoming data is written in a block indicated by the pointer.

Beside its uniform write distribution and capability in preventing heat accumulation, another main feature of TA-LRW policy is its simple yet effective victim block selection process. Despite the popularity and widespread use of the

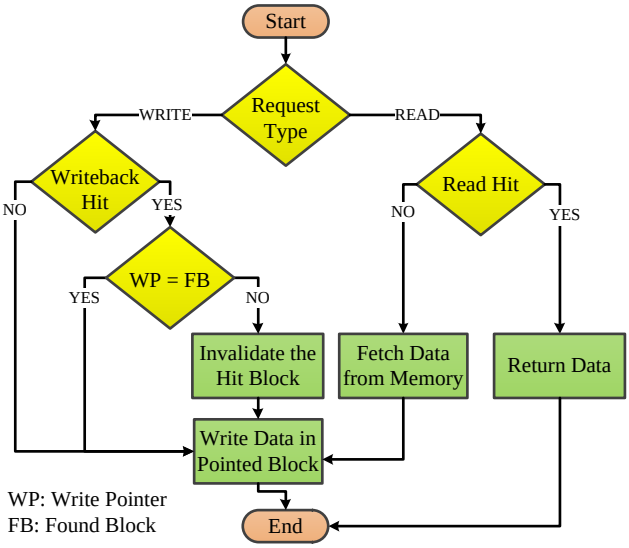


Fig. 7. Cache operation on an access request based on TA-LRW policy.

LRU policy due to a high performance provided, its complicated controlling operations and the required peripherals limit its applicability in high associative caches [46], [47]. TA-LRW policy provides a near-LRU performance with a very simple and low-cost controller.

TA-LRW sorts the blocks in a set based on their write time and evicts the oldest block on a cache miss. Both LRU and TA-LRW policies select the victim block based on the age of the blocks. While LRU policy updates the ages for every read/write accesses, TA-LRW policy updates them only for every write accesses. The oldest block in LRU policy is the least-recently read/written block and in TA-LRW policy is the least-recently written one.

Since the accesses to cache blocks are a mix of read and write operations, sorting the blocks based on their write sequence is a suitable representative for sorting them based on their read/write sequence. More clearly, the least-recently written block is highly probable to be among the elder blocks, and therefore TA-LRW policy likely evicts an old block, if not the oldest one, on a cache miss. On the other hand, recently written blocks (younger blocks in TA-LRW policy) are highly probable to be among the recently accessed blocks (younger blocks in LRU policy). Therefore, similar to LRU policy, they are not evicted from the cache on a cache miss but gotten older in TA-LRW policy. Fig. 7 depicts the TA-LRW decision according to the type of the cache access request.

In summary, the age of blocks in TA-LRW policy is determined the same as that in LRU policy except that the read accesses have no effect on the ages. The detailed results will be given in Section 4.4 to show that the age of blocks in TA-LRW policy is very close to that in LRU policy. Therefore, TA-LRW policy provides semi-LRU functionality with much simpler operations. Unlike LRU policy, which requires an age field per cache block and a complicated logic to manage these ages per access, the main component in TA-LRW policy is write pointer per cache set, which can be implemented simply in two ways shown in Fig. 8. The first implementation is shown in Fig. 8(a) in which the write pointer is updated according to TA-LRW write sequence permutation. The second implementation is to shuffle the

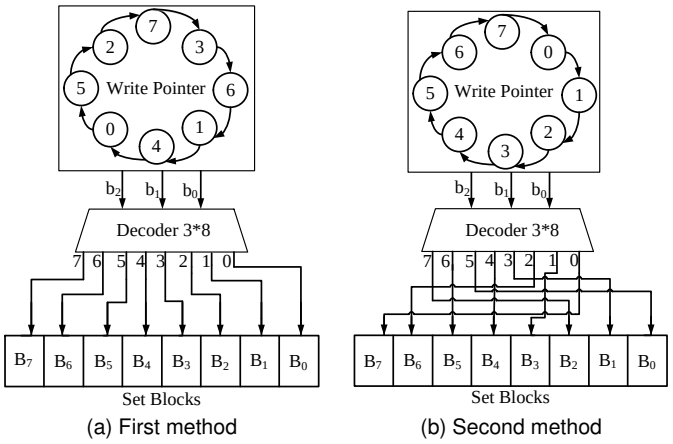


Fig. 8. Two implementations of TA-LRW pointer to select the suitable block to be written: (a) updating pointer according to TA-LRW permutation, (b) connecting decoder output to blocks according to TA-LRW permutation.

output of way selection decoder according to the TA-LRW write permutation, as depicted in Fig. 8(b).

4.4 Selection of Proper Victim Blocks

The primary goal of cache replacement policies is to provide higher performance by evicting the block that is predicted not to be needed for a longer time in the future. This prediction in both LRU and TA-LRW policies is based on the recent history of accessing to blocks.

LRU policy always evicts the oldest block (the least-recently read/written) on a cache miss. TA-LRW policy has an approximation on the access history for the block, as it only keeps track of write access history. The performance overhead of TA-LRW depends on the accuracy of this approximation. TA-LRW will be closer to LRU in term of performance by discarding the elder blocks on a cache miss. The ages of discarded blocks, based on their recent accesses, in TA-LRW policy demonstrate how close this policy is to LRU policy. Fig. 9 shows the ages of total blocks discarded by TA-LRW policy for all workloads. The age of a block can be any value from zero to seven in our 8-way associative cache.

On average, 89.5% of discarded blocks are the oldest one in a set and the block selected by TA-LRW policy is the least-recently used one. Therefore, for an average of 89.5%, TA-LRW policy decision is the same as that of LRU policy. TA-LRW policy discards the second oldest blocks (blocks with age 6) for 4.6% of cache misses, on average, and 1.5% of evictions are on blocks with age 5. Therefore, an average of 96.6% of all discarded blocks in TA-LRW policy is either the oldest block or among these elder blocks in a set.

TA-LRW policy discards the youngest block for only 0.5%, on average, for all workloads and only 0.8% of discarded blocks are the second youngest block, on average. For 3.1% of cache misses, the age of discarded blocks is two, three, or four. As observed, the majority of victim blocks selected by TA-LRW policy are old enough and suitable for eviction. Therefore, it can be expected that the impact of the proposed replacement policy on the cache miss rate and performance is minimal.

TABLE 2
Details of processor and caches configuration

CPU	arm_detailed, 1GHz, out-of-order
L1 Cache	32KB, 4-way associative, 64B block size, 2ns read and write access time, non-blocking, write-back, SRAM
L2 Cache	1MB, 8-way associative, 64B block size, 10ns read access time, 20ns write access time, non-blocking, write-back, STT-MRAM

5 Simulation Setup and Results

To evaluate the proposed TA-LRW policy, we implement it in gem5 cycle-accurate simulator [29] and use SPEC CPU2006 benchmark suite [28] as the workloads. TA-LRW policy is employed in the L2-cache of our modeled processor. The processor includes a dedicated 4-way instruction and data L1-cache and a shared 8-way L2-cache. The configuration details are given in Table 2. For each workload, the first one billion instructions are skipped as warm-up phase and the results are extracted from the next one billion instructions.

TA-LRW policy is compared with the well-known LRU policy in terms of write distribution, heat accumulation, error rate, and performance. To better illustrate how the performance of the simple TA-LRW policy approximates that of the complicated LRU policy, we also reports the performance of the FIFO policy, as a replacement policy similar to TA-LRW in terms of implementation cost and complexity.

5.1 Write Distribution

The distance between each two consecutive write operations in TA-LRW policy is three, four, or five blocks. As mentioned in Section 4, these distances are for three, two, and three out of eight consecutive write operations, respectively, for each round of writing all eight blocks in a set. Therefore, it can be predicated that 37.5% of incoming data blocks are written by three blocks away from the previous write in a set. The write distance for another 25.0% of incoming data is four blocks and the remaining 37.5% are written in blocks that are away from the previously written block by five blocks.

These values are the same for all workloads and are independent of the cache access pattern. On the other hand, the distance between two consecutive writes in LRU can be any value from zero to seven. In the worst case, this distance is zero and the most-recently written (hottest) block is written again, leading to high accumulation of heat.

In the case of one-block distance, the currently written block is adjacent to the hottest block and again leads to high heat accumulation. On the contrary, for large values of the write distances, e.g., five, six, and seven, two consecutive writes are so far to each other that almost no heat is accumulated. The heat accumulation in LRU policy depends on the contribution of write distances in total write accesses in the cache. The higher contribution of large distances, the lower heat is accumulated. However, the results for LRU policy show that the majority of incoming data blocks are written in neighboring cache blocks.

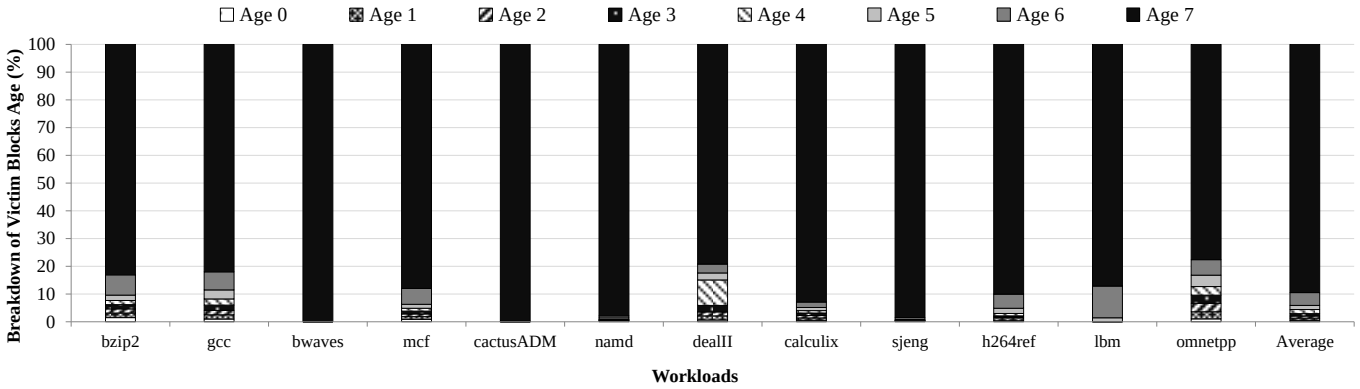


Fig. 9. Age of evicted blocks in TA-LRW replacement policy.

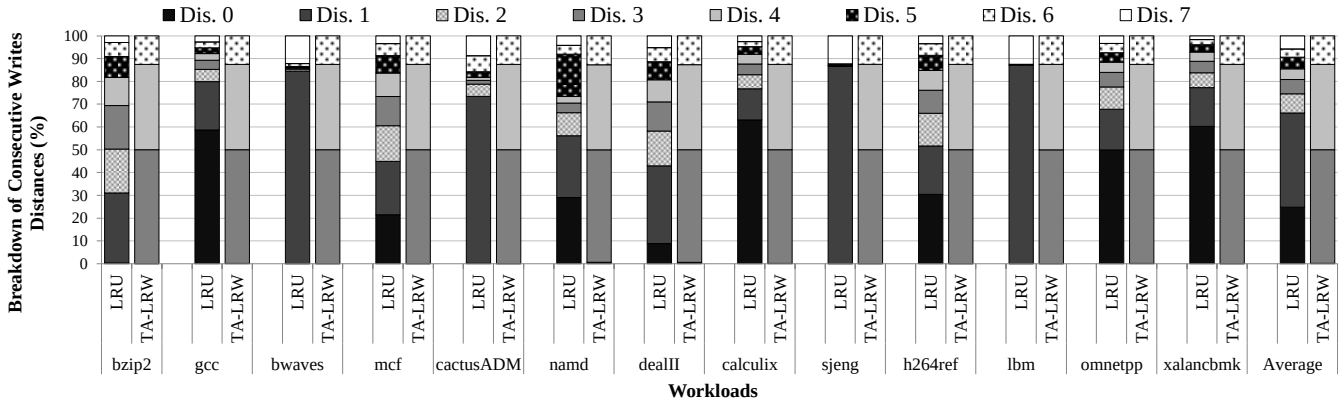


Fig. 10. Contribution of each write distance in total write operations for LRU and TA-LRW replacement policies.

Fig. 10 depicts the contribution of each write distance in total write operations for LRU and TA-LRW polices in all workloads. As shown, only distances of three, four, and five blocks contribute in TA-LRW policy by 37.5%, 25.0%, and 37.5%, respectively. While TA-LRW policy guarantees the minimum distance of three blocks, the write distance for all workloads in LRU policy is less than three blocks for more than 50.0% of write operations. On average, 24.8% of write operations are performed in a block similar to the previous write and the distance of 41.3% of write operations is only one block. Therefore, more than 66% of write operations in LRU policy are performed in the hottest block or its adjacent blocks.

5.2 Heat Accumulation

To calculate the accumulated heat for write operations, we need to indicate the temperature increase of the target block and its neighboring blocks based on the distances from the write location. Our experiments using CFD tool show that the temperature of the written block increases by 9°K just after committing the write operation. Afterward, the temperature of the neighboring blocks in the target set and its adjacent sets is increased according to Table 3, considering the equation presented in [48] for radial heat transfer in silicon nanolayers. On the other hand, the temperature of these blocks cools down toward their surrounding area afterward according to Fig. 11 based on Newton's law of cooling.

Fig. 12 shows the amount of temperature increase due to heat accumulation in cache sets. For the sake of visibility, we depicted the temperature increase for an interval of 200 successive writes in each cache set, and the results of only three workloads in LRU and TA-LRW policies are given in Fig. 12(a)-(f) [3].

As can be seen, there is a large diversity in the pattern of heat accumulation in LRU policy for the given workloads and time intervals. TA-LRW policy not only reduces the peak temperature increase to less than half of its value in

TABLE 3
Temperature increase in cache blocks based on their distance from write location

Write Distances	0	1	2	3	4	5	6	7
Increased Temperature (°K)	9.00	6.30	4.50	2.71	1.81	0.91	0.46	0.10

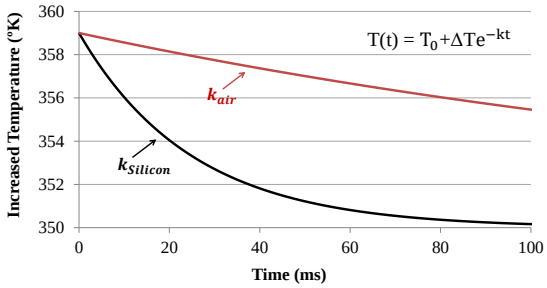


Fig. 11. Cooling down of a heat accumulated block to base temperature of the cache over the time.

3. The results for the other benchmarks are given in Fig. 17 in the Appendix Section.

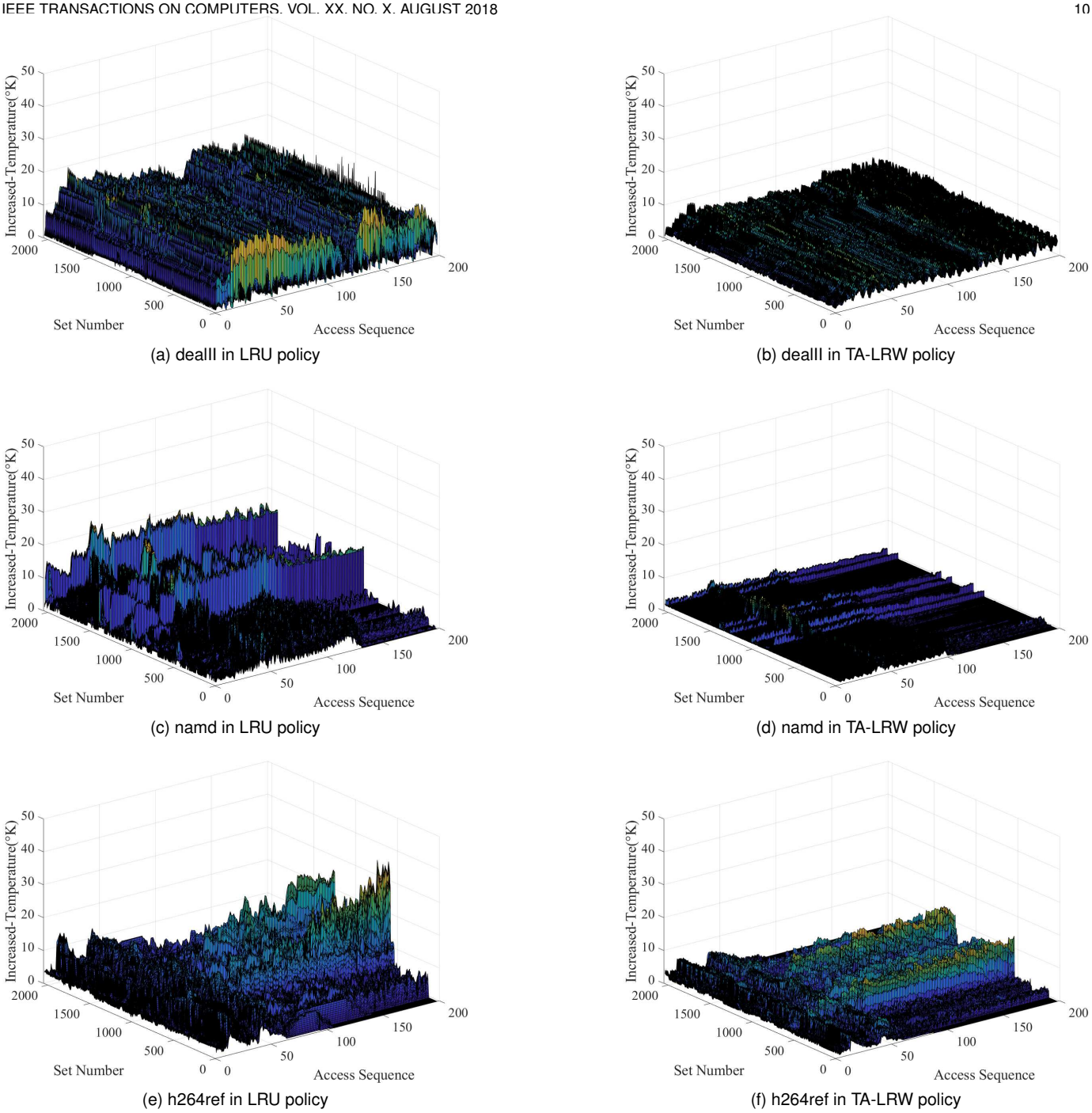


Fig. 12. Temperature increase in blocks of STT-MRAM cache in (a, b) dealII, (c, d) namd, and (e, f) h264ref workloads in comparison with LRU and TA-LRW replacement policy, respectively.

LRU policy, but also significantly mitigates the variations in cache regions temperature. Fig. 12(a) shows the increased temperature for *dealII* workload. For the majority of cache sets, the temperature is increased by 20°K in a large fraction of accesses. TA-LRW significantly decreases this value to less than 10°K, as shown in Fig. 12(b). As depicted in Fig. 12(c) for *namd* workload in LRU policy, a large diversity of temperature increase is observed in different cache sets. While the increased temperature of some sets is permanently about 20°K, this value fluctuates around 10°K for a large number of sets and is less than 5°K in some other sets. By employing TA-LRW, Fig. 12(d) illustrates that the worst-case temperature increase is less than 5°K and the variations among different sets and access sequences are not considerable. According to Fig. 12(e), the temperature of almost all cache sets is increased by around 10°K in the initial accesses of *h264ref* workload. This value gradually increases to 20°K for some sets and reaches to 35°K by the end of access sequences. By distributing the write accesses in cache sets, TA-LRW reduces the increased temperature to less than 10°K for almost all accesses in all cache sets (Fig. 12(f)). For the majority of cache sets, this value is reduced to less than 5°K.

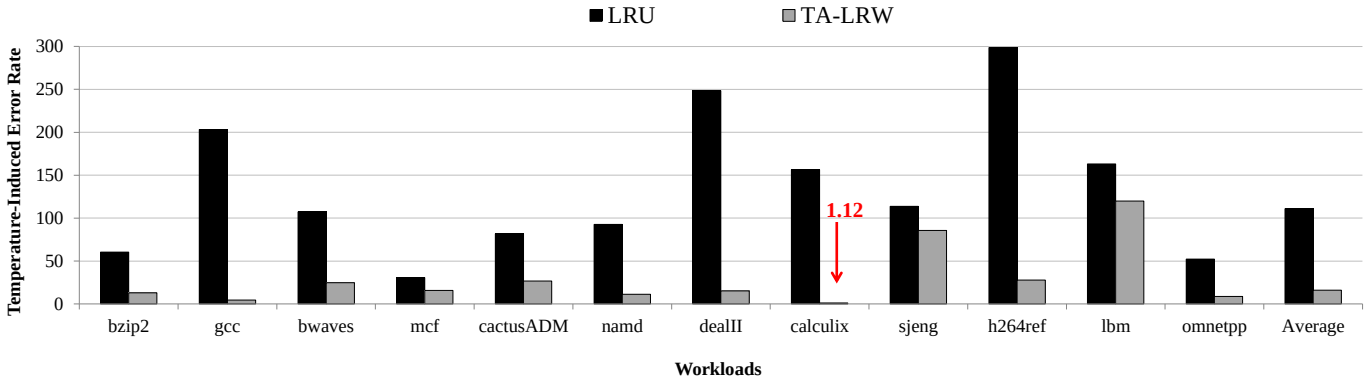


Fig. 13. Temperature-induced error rate in LRU and TA-LRW replacement policies. The values are normalized to intrinsic error rate in each workload.

5.3 Error Rate Evaluation

As mentioned, heat accumulation in the cache increases the cache error rate. A cache operating in a given temperature with no heat accumulation has an *intrinsic error rate*. Heat accumulation increases the cache error rate to a value higher than its intrinsic error rate. We define the difference between these two error rates as *temperature-induced error rate*. Intrinsic error rate is the summation of retention failure, write failure, and read disturbance for a cache operating in the base temperature. The cache is in its base temperature when no heat accumulation due to consecutive writes is considered. *Total failure rate* is the summation of retention failure, write failure, and read disturbance rates when the heat accumulation during the execution time in different parts of the cache is considered. The equations in Section 3 show the dependency of the three STT-MRAM error types to cells temperature. The transient temperature increase in the cache due to heat accumulation leads to variable rate of each error in different cache parts as well as different time intervals. The minimum values for these variable error rates are observed in the cache parts operating in the base temperature with no heat accumulation. However, the heat accumulation is observed in most parts whose temperature increases according to the amount of heat. In this case, the temperature-induced error rate is the increased error rate in the cache due to the heat accumulation effects.Therefore, the total error rate of the cache is the summation of intrinsic and temperature-induced error rates, as given in (6).

$$Total\ error\ rate = (intrinsic\ error\ rate) + (temperature\text{-}induced\ \ error\ \ rate) \quad (6)$$

Intrinsic error rate exists in the cache regardless of the replacement policy used in the cache. The goal of TA-LRW replacement policy is to minimize the temperature-induced error rate, which is a considerable fraction of the total error rate in LRU policy. Fig. 13 depicts the temperature-induced error rate in the cache for all workloads normalized to the intrinsic error rate. On average, the temperature-induced error rate in LRU policy is about 110.9% and is reduced to about 16.1% in TA-LRW. This observation indicates that TA-LRW reduces the temperature-induced error rate by 6.9x.

Temperature-induced error rate in some workloads, e.g., *gcc*, *dealII*, and *h264ref*, is more than 200% in LRU policy. This value indicates that heat accumulation can increase the total error rate in the cache by more than twice. The minimum value of the temperature-induced error rate in LRU is about 30.5% for *mcf*. For TA-LRW, the temperature-induced error rate is less than 30% for all workloads except in *sjeng* and *lbm* workloads. These values are 85.6% and 119.9% for *sjeng* and *lbm* workloads, respectively, which are still much lower than that in LRU.

The proposed TA-LRW policy aims to minimize the heat accumulation and its functionality is independent of external parameters, e.g., cores activity, transactions on buses, and cooling system. In this regard, we assume that external parameters affect the cache temperature evenly and provide a uniform base temperature for all cache regions. The heat accumulation challenge is the difference between cache momentary temperature and its base temperature. The main aim of TA-LRW policy is to minimize the gap between the base temperature and the cache momentary temperature. Heat accumulation due to consecutive writes remains a major challenge without any dependency on STT-MRAM cache base temperature, its diversity during time, cooling system functionality, and the uniformity or non-uniformity of heat transition. The consequence of heat accumulation is the increase in heat-sensitive cache error rate, which is mitigated by TA-LRW.

5.4 Performance Evaluation

Fig. 14 depicts the L2 cache miss rate in both LRU and TA-LRW policies for all workloads. To better illustrate how TA-LRW policy is close to LRU policy in term of performance, we have also included the miss rate of FIFO policy in Fig. 14. As mentioned, the complexity and implementation cost of TA-LRW is almost the same as that of FIFO. Hence, it may be misinterpreted that their strategy on victim block selection is almost the same, resulting in similar performance. However, this is not the case and the proposed FIFO-like replacement policy in term of complexity, is close to LRU in term of performance. Including the results of FIFO policy provides a better insight into the efficiency of TA-LRW policy by demonstrating how far is FIFO policy from LRU and TA-LRW policies in term of performance.

According to Fig. 14, TA-LRW increases the miss rate by an average of 2.9%, compared with LRU policy. This value for FIFO policy is as high as 9.5%. For some workloads, e.g., *bwaves*, *cactusADM*, *namd*, and *sjeng*, the miss rate in both FIFO and TA-LRW is slightly lower than that in LRU. The miss rate of TA-LRW in *bzip2* workload is almost the same as that in LRU, whereas FIFO significantly increases the

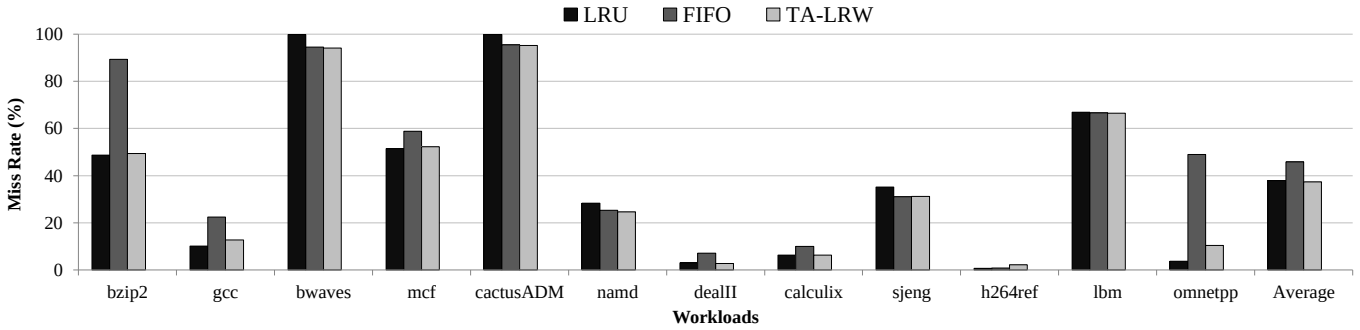


Fig. 14. Miss Rate of TA-LRW in comparison with LRU and FIFO replacement policies.

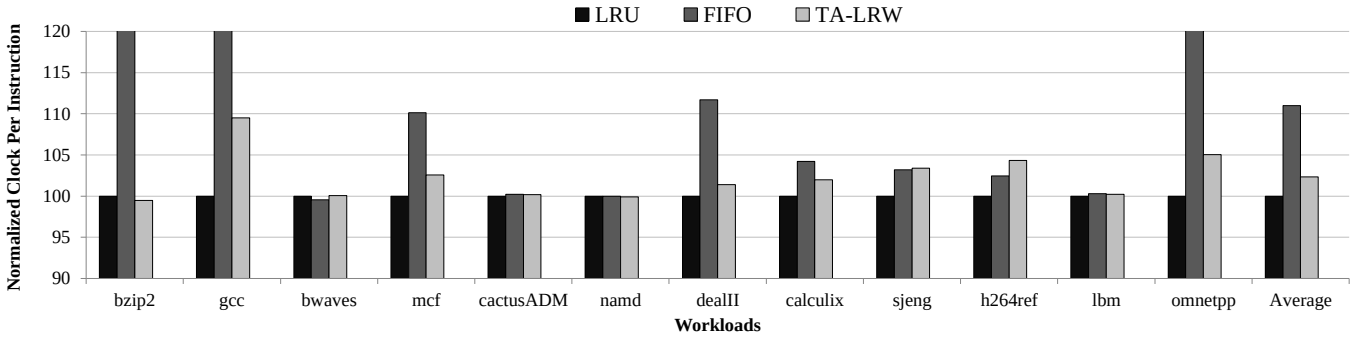


Fig. 15. Clock Per Instruction (CPI) in FIFO and TA-LRW replacement policies normalized to CPI in LRU replacement policy.

miss rate. In *omnetpp* workload, TA-LRW increases the miss rate by 6.2% and this increase in FIFO is 45.1%. FIFO policy increases the miss rate in *mcf*, *dealII*, and *calculix* workloads, while this rate in TA-LRW is almost the same as that in LRU. The difference in L2 cache miss rate by the evaluated replacement policies affects the performance of the system.

We consider $Cycles\ Per\ Instruction$ (CPI) as the performance metric and depict the CPI for all three policies in Fig. 15. On average, the performance overhead in FIFO is 10.3%, compared with LRU policy. However, TA-LRW increases the CPI by only 2.3%, on average. FIFO increases the CPI by more than 20% in *bzip2*, *gcc*, and *omnetpp* workloads, whereas the worst case overhead of TA-LRW is 9.3%, which is observed in *gcc* workload. The performance of all evaluated replacement policies are almost the same in some workloads, e.g., *bwaves*, *cactusADM*, *namd*, and *lbm*. There is only two workloads, i.e., *sjeng* and *h264ref*, in which the CPI of TA-LRW is slightly higher than that of FIFO. TA-LRW outperforms LRU in *bzip2* workload, in which the performance overhead of FIFO is more than 50%.

In summary, TA-LRW offers a near-LRU performance without requiring the complicated LRU components and circuitry. The implementation of TA-LRW is also as simple as FIFO, while providing a much better performance. Considering an N-way associative cache, LRU policy needs to add $\log_2 N$ extra bits to each cache block for sorting the age values. Therefore, the number of extra bits in each cache set is $N \log_2 N$. The extra bits in each cache set required by TA-LRW is only $\log_2 N$ (same as in FIFO), which is N times lower than that in LRU. Moreover, LRU conducts a set of operations to update the age bits of blocks for each cache access and to find the suitable victim for each cache miss. TA-LRW only updates the pointer of accessed set for each write access. Due to its smaller components and simpler operations, replacing LRU policy with TA-LRW policy significantly reduces the area and energy consumption of the cache replacement policy, in addition to uniformly distributing the generated heat in the cache.

It is noteworthy that TA-LRW is the first, but not necessary the optimal thermal-aware replacement policy for STT-MRAM caches. Utilizing a more precise thermal model for the cache on which the replacement policy makes decisions can help for a better heat distribution. Besides, including a trade-off between miss rate and temperature can provide a higher performance. This study will spark further research in designing more advanced thermal-aware replacement policies for STT-MRAM caches.

6 Related Work and Discussion

There are several studies addressing the reliability of STT-MRAM caches and presenting techniques to overcome either write failure, read disturbance, or retention failure. These techniques, in the best case, reduce the rate of one error type without affecting the other error types. However, most of them adversely affect the rate of other error types to alleviate the target error type.

To reduce the rate of write failure, some studies increase the amplitude and/or width of write pulse [49]–[51]. In addition to imposing high latency and energy overhead, these techniques increase the probability of oxide barrier breakdown in STT-MRAM cells. In studies presented in [52], [53], the content of memory is read after each write operation and the write operation is repeated until the data is cor-

rectly written. Extra read operations increase the occurrence probability of read disturbance in these techniques.

Error-Correcting Codes (ECCs) have been widely used to tolerate write failures [5], [37], [53]–[58]. These codes increase the read disturbance rate as well as the cache area and energy consumption. Some studies try to reduce the number of bit switching in write operations by encoding the incoming data or writing into a cache block with the minimum hamming distance [43], [55], [59]. These techniques complicate the design and increase the read disturbance rate by imposing extra read operations.

To reduce the read disturbance rate, read current is decreased in studies presented in [34], [60]. To correctly read the contents of memory cells at reduced read current, these studies present more accurate sense amplifiers. *Destructive read and restoring* technique performs an extra write operation after each read operation to correct possible read disturbance error [61], [62]. This technique not only imposes energy consumption overhead, but also significantly increases the write failure rate. The number of restore operations is reduced in [18] by selectively restore the erroneous blocks. Using ECC is another approach to correct read disturbance errors, which imposes area, performance, and energy overhead [57].

To reduce the retention failure rate, some studies suggested to increase the STT-MRAM cell retention time by manipulating the MTJ layers characteristics or the cell thermal stability factor [21], [34], [52], [63], [64]. These changes in STT-MRAM characteristics increase the write failure rate. Employing ECC is another approach for overcoming the retention failure rate [39]. Same as the side effects of ECC in tolerating read disturbance and write failure, ECC increases the occurrence probability of these two error types in addition to imposing energy consumption and area overhead.

Refreshing the STT-MRAM cells is another technique to reduce the retention failure rate [65]. However, refreshing is beneficial only if assuming a deterministic retention time for all of the memory cells, which is not the case as the retention failure occurs stochastically. Considering some error correction mechanisms, e.g., using ECCs, memory scrubbing is suggested in [66] to periodically check/correct the cache contents and prevent the error accumulation.

Some studies reduce the write energy and latency by decreasing the write current and/or pulse width, which leads to increase in write failure rate [11], [67], [68], [68], [69]. To moderate this increase, these studies decrease STT-MRAM thermal stability factor (Δ). As its side effect, read disturbance and retention failure rates are exponentially increased. Reduction in write energy decreases the temperature-induced rates of all three error types on the one hand and significantly increases the intrinsic rates of read disturbance and retention failure on the other hand. The total error rate is highly increased in this case. As an example, our investigations show that temperature-induced error rate due to heat accumulation increases the total error rate to 2x, which fades out by using smaller write current or write pulse width. However, reduction of Δ from 45 to 35 will increase the total error rate by four orders of magnitude. Therefore, the total error rate of STT-MRAM cache significantly increases by reducing the write current and thermal stability factor, which makes this approach useless for error rate reduction.

Besides the mentioned schemes in emerging STT-MRAM caches, an extensive effort is conducted for decades to improve the reliability of SRAM caches [70]–[72]. Despite the similar functionality of SRAM and STT-MRAM cache, reliability improvement schemes of SRAM caches are generally inapplicable or inefficient in STT-MRAM caches. This is due to the difference in source and pattern of errors in these two technologies.

The main source of errors in SRAM caches is radiation-induced particle strike, which causes bit-flip in a single or multiple adjacent memory cells with exponential probability distribution [73]–[75]. Recent studies have shown that STT-MRAM cells are immune to this source of errors. On the other hand, retention failure, write failure, and read disturbance errors in STT-MRAM caches are originated from thermal instability and stochastic behavior of the cells, which is not a concern in SRAM cells. Beside STT-MRAM physical parameters, the occurrence probability and pattern of these errors depends on the cache access behavior and the data content. Because of these differences, not only the reliability improvement schemes in SRAM and STT-MRAM caches are incompatible, but also it is not easy to provide a fair comparison between their reliability. The benefits of replacing SRAM with STT-MRAM are mainly due to non-volatility, near-zero leakage power, and higher density of STT-MRAM, whereas its reliability is among the major challenges for its commercialization. This study is one step ahead to overcoming the drawbacks of STT-MRAM caches.

To the best of our knowledge, TA-LRW is the first technique that simultaneously mitigates the rates of retention failure, read disturbance, and write failure in STT-MRAM caches. As shown in Section 5, TA-LRW is an effective approximation of conventional LRU replacement policy, with significantly simpler design. As observed in Fig. 15, the performance of TA-LRW policy is almost the same as that in LRU policy. The only challenge in TA-LRW policy is its performance overhead in a rarely-occurred scenario in which a high fraction of victim blocks in cache misses are not among the old blocks. This scenario is probable when the write history of blocks is not an accurate approximation for their access history. This inaccuracy increases only when a large fraction of cache accesses is due to read in elder blocks in the cache, which moves the old blocks to the head of LRU queue. However, our evaluations indicate that the mentioned scenario is a rare event.

7 Conclusions

Emerging STT-MRAM on-chip caches are highly error-prone due to stochastic characteristics of this memory technology. The error rate in cache memory increases exponentially in higher temperature. Our investigation shows that the temporal locality in writing to adjacent cache blocks causes regional heat accumulation inside the cache. To prevent this heat accumulation, we proposed *Thermal-Aware Least-Recently Written* (TA-LRW) replacement policy. TA-LRW policy keeps track of write operations into cache blocks and replaces the blocks based on their write history on a cache miss. The proposed replacement policy guarantees that the distance of two consecutive write operations in a set is at least three blocks. Regarding the advantage of TA-LRW over

conventional replacement policies in STT-MRAM LLCs, our evaluation shows that the proposed TA-LRW policy reduces the temperature-induced error rate by 94.8% compared with the conventional LRU-based LLC. In addition, the hardware complexity, energy consumption, and area of TA-LRW are much lower than those of LRU policy and its performance overhead is only 2.3%. The proposed TA-LRW replacement policy is not only an effective solution for heat distribution in STT-MRAM caches, but also an effective approximation and promising alternative for LRU policy in highly associative cache for minimizing its complexity.

ACKNOWLEDGMENTS

The authors would like to thank Ali Bijarchi, Dr. Amir Mahdi Hosseini Monazzah, and Dr. Nezam Rohbani for their contribution and discussion on setting up the STT-MRAM cell temperature simulation.

REFERENCES

[1] R. Iyer, L. Zhao, F. Guo, R. Illikkal, S. Makineni, D. Newell, Y. Solihin, L. Hsu, and S. Reinhardt, "QoS policies and architecture for cache/memory in CMP platforms," in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2007, pp. 25–36.

[2] 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, 2018.

[3] E. I. Vatajelu, P. Prinetto, M. Taouil, and S. Hamdioui, "Challenges and solutions in emerging memory testing," *IEEE Transactions on Emerging Topics in Computing (TETC)*, 2017, in press.

[4] 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 the European Dependable Computing Conference (EDCC)*, 2016, pp. 120–129.

[5] Z. Azad, , H. Farbeh, A. M. Hosseini Monazah, and S. G. Miremadi, "AWARE: Adaptive Way Allocation for Reconfigurable ECCs to protect write errors in STT-RAM caches," *IEEE Transactions on Emerging Topics in Computing (TETC)*, 2017, in press.

[6] [Online]. Available: Available:http://www.itrs.net/, [Accessed March 23, 2016].

[7] M. Imani, S. Patil, and T. Rosing, "Approximate computing using multiple-access single-charge associative memory," *IEEE Transactions on Emerging Topics in Computing (TETC)*, 2017, in press.

[8] N. Kim and K. Choi, "Exploration of trade-offs in the design of volatile stt–ram cache," *Elsevier Journal of Systems Architecture (JSA)*, vol. 71, pp. 23–31, 2016.

[9] S. P. Park, S. Gupta, N. Mojumder, A. Raghunathan, and K. Roy, "Future cache design using stt mrams for improved energy efficiency: Devices, circuits and architecture," in *Proceedings of the ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2012, pp. 492–497.

[10] J. Wang, X. Dong, and Y. Xie, "Oap: an obstruction-aware cache management policy for stt-ram last-level caches," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*. IEEE, 2013, pp. 847–852.

[11] 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 the ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2012, pp. 243–252.

[12] L. Jiang, B. Zhao, Y. Zhang, and J. Yang, "Constructing large and fast multi-level cell stt-mram based cache for embedded processors," in *Proceedings of the ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2012, pp. 907–912.

[13] 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 the IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2011, pp. 50–61.

[14] B.-M. Lee and G.-H. Park, "Performance and energy-efficiency analysis of hybrid cache memory based on sram-mram," in *Proceedings of the IEEE International SoC Design Conference (ISOCC)*, 2012, pp. 247–250.

[15] J. Choi and G.-H. Park, "Nvm way allocation scheme to reduce nvm writes for hybrid cache architecture in chip-,multiprocessors," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2017, in press.

[16] H. Farbeh, H. Kim, S. G. Miremadi, and S. Kim, "Floating-ecc: Dynamic repositioning of error correcting code bits for extending the lifetime of stt-ram caches," *IEEE Transactions on Computers (TC)*, vol. 65, no. 12, pp. 3661–3675, 2016.

[17] A. V. Khvalkovskiy, D. Apalkov, S. Watts, R. Chepulskii, R. Beach, A. E. Ong, X. Tang, A. Driskill-Smith, W. Butler, P. Visscher, D. Lottis, E. Chen, V. Nikitin, and M. Krounbi, "Basic principles of stt-mram cell operation in memory arrays," *Journal of Physics D: Applied Physics*, vol. 46, no. 7, pp. 1–35, 2013.

[18] R. Wang, L. Jiang, Y. Zhang, L. Wang, and J. Yang, "Selective restore: An energy efficient read disturbance mitigation scheme for future stt-mram," in *Proceedings of the ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[19] A. Chen, "A review of emerging Non-Volatile Memory (NVM) technologies and applications," *Elsevier Solid-State Electronics*, vol. 125, pp. 25–38, 2016.

[20] A. Chintaluri, A. Parihar, S. Natarajan, H. Naeimi, and A. Raychowdhury, "A model study of defects and faults in embedded Spin Transfer Torque (STT) MRAM arrays," in *Proceedings of the IEEE Asian Test Symposium (ATS)*, 2015, pp. 187–192.

[21] X. Bi, H. Li, and J.-J. Kim, "Analysis and optimization of thermal effect on stt-ram based 3-d stacked cache design," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2012, pp. 374–379.

[22] M. V. Beigi and G. Memik, "Tesla: Using microfluidics to thermally stabilize 3d stacked stt-ram caches," in *Proceedings of the IEEE International Conference on Computer Design (ICCD)*, 2016, pp. 344–347.

[23] X. Bi, H. Li, and X. Wang, "Stt-ram cell design considering cmos and mtj temperature dependence," *IEEE Transactions on Magnetics (TMAG)*, vol. 48, no. 11, pp. 3821–3824, 2012.

[24] R. Bishnoi, M. Ebrahimi, F. Oboril, and M. B. Tahoori, "Asynchronous asymmetrical write termination (aawt) for a low power stt-mram," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*, 2014, pp. 1–6.

[25] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3d stacked mram l2 cache for cmps," in *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2009, pp. 239–249.

[26] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen, "Circuit and microarchitecture evaluation of 3d stacking magnetic ram (mram) as a universal memory replacement," in *Proceedings of the ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2008, pp. 554–559.

[27] "Fluent 12.1.4 Documantation, User guide," Fluent, Inc., 2009, [Accessed May 11, 2016].

[28] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[29] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

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

[31] Z. Sun, X. Bi, and H. Li, "Process variation aware data management for stt-ram cache design," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2012, pp. 179–184.

[32] D. Apalkov, A. Khvalkovskiy, S. Watts, V. Nikitin, X. Tang, D. Lottis, K. Moon, X. Luo, E. Chen, A. Ong, A. Driskill-Smith, and M. Krounbi, "Spin-transfer torque magnetic random access memory (stt-mram)," *ACM Journal on Emerging Technologies in Computing Systems (JETCS)*, vol. 9, no. 2, p. 13, 2013.

[33] B. Dieny, R. B. Goldfarb, and K.-J. Lee, *Introduction to magnetic random-access memory*. John Wiley & Sons, 2016.

[34] W. Zhao, T. Devolder, Y. Lakys, J.-O. Klein, C. Chappert, and P. Mazoyer, "Design considerations and strategies for high-reliable stt-mram," *Elsevier Microelectronics Reliability (MR)*, vol. 51, no. 9, pp. 1454–1458, 2011.

[35] 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, 2016.

[36] W. Zhao, Y. Zhang, T. Devolder, J.-O. Klein, D. Ravelosona, C. Chappert, and P. Mazoyer, "Failure and reliability analysis of stt-mram," *Elsevier Microelectronics Reliability (MR)*, vol. 52, no. 9, pp. 1848–1852, 2012.

[37] Z. Azad, , H. Farbeh, A. M. Hosseini Monazah, 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, 2017.

[38] E. Eken, Y. Zhang, W. Wen, R. Joshi, H. Li, and Y. Chen, "A novel self-reference technique for stt-ram read and write reliability enhancement," *IEEE Transactions on Magnetics (TMAG)*, vol. 50, no. 11, pp. 1–4, 2014.

[39] 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, pp. 54–75, 2013.

[40] A. Chintaluri, H. Naeimi, S. Natarajan, and A. Raychowdhury, "Analysis of defects and variations in embedded spin transfer torque (stt) mram arrays," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 6, no. 3, pp. 319–329, 2016.

[41] Y. Ran, W. Kang, Y. Zhang, J.-O. Klein, and W. Zhao, "Read disturbance issue of design techniques for nanoscale STT-MRAM," *Elsevier Journal of Systems Architecture (JSA)*, vol. 71, pp. 2–11, 2016.

[42] Z. Pajouhi, X. Fong, A. Raghunathan, and K. Roy, "Yeild, area, and energy optimization in STT-MRAMs using failure-aware ECC," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 2, p. 20, 2016.

[43] A. M. Hosseini Monazah, H. Farbeh, and S. G. Miremadi, "LER: Least Error Rate replacement algorithm for emerging STT-RAM caches," *IEEE Transactions on Device and Materials Reliability (TDMR)*, vol. 16, no. 2, pp. 220–226, 2016.

[44] X. Chen, N. Khoshavi, R. F. DeMara, J. Wang, D. Huang, W. Wen, and Y. Chen, "Energy-aware adaptive restore schemes for mlc stt-ram cache," *IEEE Transactions on Computers (TC)*, vol. 66, no. 5, pp. 786–798, 2017.

[45] S. Hong, J. Lee, and S. kim, "Ternary cache: Three-valued mlc stt-ram caches," in *Proceedings of the IEEE International Conference on Computer Design (ICCD)*, 2014, pp. 83–89.

[46] T. Sudarshan, R. A. Mir, and S. Vijayalakshmi, "Highly efficient lru implementations for high associativity cache memory," in *Proceedings of the IEEE International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2004, pp. 24–35.

[47] H. Liu, M. Ferdman, J. Huh, and D. Burger, "Cache bursts: A new approach for eliminating dead blocks and increasing cache efficiency," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2008, pp. 222–233.

[48] A. Sellitto and V. Cimmelli, "Flux limiters in radial heat transport in silicon nanolayers," *Journal of Heat Transfer*, vol. 136, no. 7, p. 071301, 2014.

[49] H. Sun, C. Liu, T. Min, N. Zheng, and T. Zhang, "Architectural exploration to enable sufficient mtj device write margin for stt-ram based cache," *IEEE Transactions on Magnetics (TMAG)*, vol. 48, no. 8, pp. 2346–2351, 2012.

[50] Y. Emre, C. Yang, K. Sutaria, Y. Cao, and C. Chakrabarti, "Enhancing the reliability of stt-ram through circuit and system level techniques," in *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS)*, 2012, pp. 125–130.

[51] Y. Lakys, W. S. Zhao, T. Devolder, Y. Zhang, J.-O. Klein, D. Ravelosona, and C. Chappert, "Self-enabled error-free switching circuit for spin transfer torque mram and logic," *IEEE Transactions on Magnetics (TMAG)*, vol. 48, no. 9, pp. 2403–2406, 2012.

[52] H. Sun, C. Liu, N. Zheng, T. Min, and T. Zhang, "Design techniques to improve the device write margin for MRAM-based cache memory," in *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*, 2011, pp. 97–102.

[53] X. Bi, Z. Sun, H. Li, and W. Wu, "Probabilistic design methodology to improve run-time stability and performance of stt-ram caches," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, 2012, pp. 88–94.

[54] Y. Zhang, X. Wang, Y. Li, A. K. Jones, and Y. Chen, "Asymmetry of mtj switching and its implication to stt-ram designs," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*, 2012, pp. 1313–1318.

[55] W. Wen, M. Mao, X. Zhu, S. H. Kang, D. Wang, and Y. Chen, "CD-ECC: Content-Dependent Error Correction Codes for combating asymmetric nonvolatile memory operation errors," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 1–8.

[56] J. Ahn, S. Yoo, and K. Choi, "Selectively protecting error-correcting code for area-efficient and reliable STT-RAM caches," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2013, pp. 285–290.

[57] S. M. Seyedzadeh, R. Maddah, A. Jones, and R. Melhem, "Leveraging ecc to mitigate read disturbance, false reads and write faults in stt-ram," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2016, pp. 215–226.

[58] Z. Azad, H. Farbeh, and A. M. H. Monazzah, "Orient: Organized interleaved eccs for new stt-mram caches," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*. IEEE, 2018, pp. 1187–1190.

[59] R. Maddah, S. M. Seyedzadeh, and R. Melhem, "CAFO: Cost Aware Flip Optimization for Asymmetric Memories," in *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, 2015, pp. 320–330.

[60] T. Na, J. P. Kim, S. H. Kang, and S.-O. Jung, "Read disturbance reduction technique for offset-canceling dual-stage sensing circuits in deep submicrometer stt-ram," *IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS-II)*, vol. 63, no. 6, pp. 578–582, 2016.

[61] R. Takemura, T. Kawahara, K. Ono, K. Miura, H. Matsuoka, and H. Ohno, "Highly-scalable disruptive reading scheme for gb-scale spram and beyond," in *Proceedings of the IEEE International Memory Workshop (IMW)*, 2010, pp. 1–2.

[62] L. Jiang, W. Wen, D. Wang, and L. Duan, "Improving read performance of stt-mram based main memories through smash read and flexible read," in *Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2016, pp. 31–36.

[63] 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 perspective," *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems*, vol. 18, no. 12, pp. 1710–1723, 2010.

[64] A. Driskill-Smith, "Latest advances and future prospects of stt-ram," in *Proceedings of Non-Volatile Memories Workshop (NVMW)*, 2010, pp. 11–13.

[65] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient stt-mram caches," in *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, 2011, pp. 50–61.

[66] S. Mittal, "A survey of soft-error mitigation techniques for non-volatile memories," *Computers*, vol. 6, no. 1, p. 8, 2017.

[67] F. Arezoomand, A. Asad, M. Fazeli, M. Fathy, and F. Mohammadi, "Energy aware and reliable stt-ram based cache design for 3d embedded chip-multiprocessors," in *Proceedings of International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2017, pp. 1–8.

[68] Z. Sun, X. Bi, H. Li, W.-F. Wong, Z.-L. Ong, X. Zhu, and W. Wu, "Multi retention level stt-ram cache designs with a dynamic refresh scheme," in *Proceedings of the International Symposium on Microarchitecture (MICRO)*, 2011, pp. 329–338.

[69] K. Kuan and T. Adegbija, "Lars: Logically adaptable retention time stt-ram cache for embedded systems," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*, 2018, pp. 1–6.

[70] M. Zhang, V. M. Stojanovic, and P. Ampadu, "Reliable ultra-low-voltage cache design for many-core systems," *IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS-II)*, vol. 59, no. 12, pp. 858–862, 2012.

[71] H. Farbeh and S. G. Miremadi, "Psp-cache: A low-cost fault-tolerant cache memory architecture," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*, 2014, pp. 1–4.

[72] M. Manoochehri, M. Annavaram, and M. Dubois, "Cppc: correctable parity protected cache," in *roceedings of the annual international symposium on Computer architecture (ISCA)*, vol. 39, no. 3, 2011, pp. 223–234.

[73] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe, "Multi-bit error tolerant caches using two-dimensional error coding,"

in *Proceedings of the International Symposium on Microarchitecture (MICRO)*, 2007, pp. 197–209.
[74] H. Farbeh, N. S. Mirzadeh, N. F. Ghalaty, S.-G. Miremadi, M. Fazeli, and H. Asadi, "A cache-assisted scratchpad memory for multiple-bit-error correction," *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems*, vol. 24, no. 11, pp. 3296–3309, 2016.
[75] A. Neale, M. Jonkman, and M. Sachdev, "Adjacent-mbu-tolerant sec-ded-taec-yaed codes for embedded srams," *IEEE Transactions on Circuits and Systems II: (TCAS-II)*, vol. 62, no. 4, pp. 387–391, 2015.

Elham Cheshmikhani received the B.Sc. degree in computer engineering from Iran University of Science and Technology (IUST) and the M.Sc. degree in computer engineering from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2011 and 2013, respectively. She is currently a PhD candidate in computer engineering at Sharif University of Technology (SUT), Tehran, Iran. She was a member of 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.

Hamed Farbeh (S'12) received the B.Sc., M.Sc., and PhD degrees in computer engineering from Sharif University of Technology (SUT), Tehran, Iran, in 2009, 2011, and 2017, respectively. He was been a member of the Dependable Systems Laboratory (DSL) at SUT from 2007 to 2017 and the head of DSL from April 2017 to February 2018. He is currently the faculty member of the Department of Computer Engineering and Information Technology, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. He was with the Embedded Computing Laboratory (ECL), KAIST, Daejeon, South Korea, as a Visiting Researcher from October 2014 to May 2015 and collaborated with the Institute of Research for Fundamental Sciences (IPM), Tehran, Iran, as Postdoc fellow from May 2017 to January 2018. His current research interests include reliable memory hierarchy, reliability challenges in emerging memory technologies, cyber-physical systems. He was the IEEE student member from 2012 to 2017.

Seyed Ghassem Miremadi (SM'07) received the M.Sc. degree in applied physics and electrical engineering from the Linkping Institute of Technology, Linkping, Sweden, and the Ph.D. degree in computer engineering from the Chalmers University of Technology, Gothenburg, Sweden.

He initiated the Dependable Systems Laboratory with the Sharif University of Technology (SUT), Tehran, Iran, in 1996, as fault-tolerant computing is his specialty, and has chaired the laboratory since then. He and his group have done research in physical, simulation-based and software-implemented fault injection, dependability evaluation using HDL models, fault-tolerant embedded systems, fault-tolerant networks-on-chip, fault-tolerant real-time systems, and fault-tolerant storage systems. He was the Education Director from 1997 to 1998, the Head from 1998 to 2002, the Research Director from 2002 to 2006, and the Director of the Hardware Group with the Computer Engineering Department, SUT, from 2009 to 2010. From 2003 to 2010, he was the Director of the Information Technology Program with SUT International Campus-Kish Island, Kish, Iran. From 2010 to 2012 and since 2014, he has been the Vice President of Academic Affairs at SUT, where he is currently a Professor of Computer Engineering. Dr. Miremadi is a Senior Member of the IEEE Computer Society and the IEEE Reliability Society. He served as the General Co-Chair of the 13th International CSI Computer Conference in 2008, the Executive Chair of the International Conference on Engineering Education in 2013, and the General Co-Chair of the International CSI Symposium on Real-Time and Embedded Systems and Technologies in 2015. He is the Editor of the *Scientia Transactions on Computer Science and Engineering*.

Hossein Asadi (M'08, SM'14) received the B.Sc. and M.Sc. degrees in computer engineering from the SUT, Tehran, Iran, in 2000 and 2002, respectively, and the Ph.D. degree in electrical and computer engineering from Northeastern University, Boston, MA, USA, in 2007. He was with EMC Corporation, Hopkinton, MA, USA, 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 tenured Associate Professor. He is the Founder and Director of the Data Storage, Networks, and Processing (DSN) Laboratory, Director of Sharif High-Performance Computing Center, the Director of Sharif Information Technology Service Center (ITC), and the President of Sharif ICT Innovation Center. He spent three months in the summer 2015 as a Visiting Professor at the School of Computer and Communication Sciences at the Ecole Poly-technique Federele de Lausanne (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 seventy technical papers in reputed journals and conference proceedings. His current research interests include data storage systems and networks, solid-state drives, operating system support for I/O and memory management, and 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 Internation Symposium on Computer Architecture and 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, and the Distinguished Technology Award from SUT in 2017. He is also recipient of Extraordinary Ability in Science visa from US Citizenship and Immigration Services in 2008. He has also served as the publication chair of several national and international conferences including CNDS2013, AISP2013, and CSSE2013 during the past four years. Most recently, he has served as a Guest Editor of IEEE Transactions on Computers, a Program Co-Chair of the 18th International Symposium on Computer Architecture & Digital Systems (CADS2015), and the Program Chair of CSI National Computer Conference (CSICC2017).

APPENDIX

A. ACCUMULATED WRITE DISTANCES IN LRU POLICY

To show the breakdown of write distances for all workloads, the results for the remaining workloads are shown in Fig. 16. Write locality in *bwaves* workload shows that less than 30% of data is written in the blocks with more than 3 distances, as depicted in Fig. 16(a). The same behavior is observed in *h264ref*, *omnetpp*, and *mcf* workloads (Fig 16(b), (e) and (f), respectively). Fig. 16(c) indicates that for some time intervals, a large fraction of consecutive writes in *sjeng* workload is conducted in adjacent blocks. The write distance for about 40% of accesses in *namd* workload is either zero, one, or two, according to Fig. 16(g), which results in a large amount of heat accumulation. Fig. 16(h) shows that the heat accumulation in *calculix* benchmark is even higher than that in *namd* workload since the write distance is less than three for more than 50% of write accesses in the majority of time intervals.

B. HEAT ACCUMULATION IN LRU POLICY

Fig. 17 represents the increased temperature in STT-MRAM cache sets operating under LRU and TA-LRW policies. The results for 200 consecutive write accesses per cache set demonstrate a large increase and high diversity in temperature of cache sets in LRU policy for different workloads. Fig. 17(a) shows that the increased temperature for *bwaves* workload in LRU policy varies from 30°K to 50°K with the average of about 40°K. Using TA-LRW policy, the increased temperature is about 20°K with significantly lower variations compared with LRU, as shown in Fig. 17(b). According to Fig. 17(c), which depicts the LRU policy behavior in *calculix* workload, the temperature is increased by 10°K in majority of cache sets. This increase is higher than 30°K for some sets in all their access sequences. TA-LRW reduces the increased temperature to lower than 7°K for all sets in all access sequences (Fig. 17(d)).

Using LRU policy, the temperature increase in *bzip2* workload is less than 20°K for initial accesses and rapidly reaches above 30°K for all remaining accesses in all sets (Fig. 17(e)). TA-LRW limits this increase to about 10°K (Fig. 17(f)).

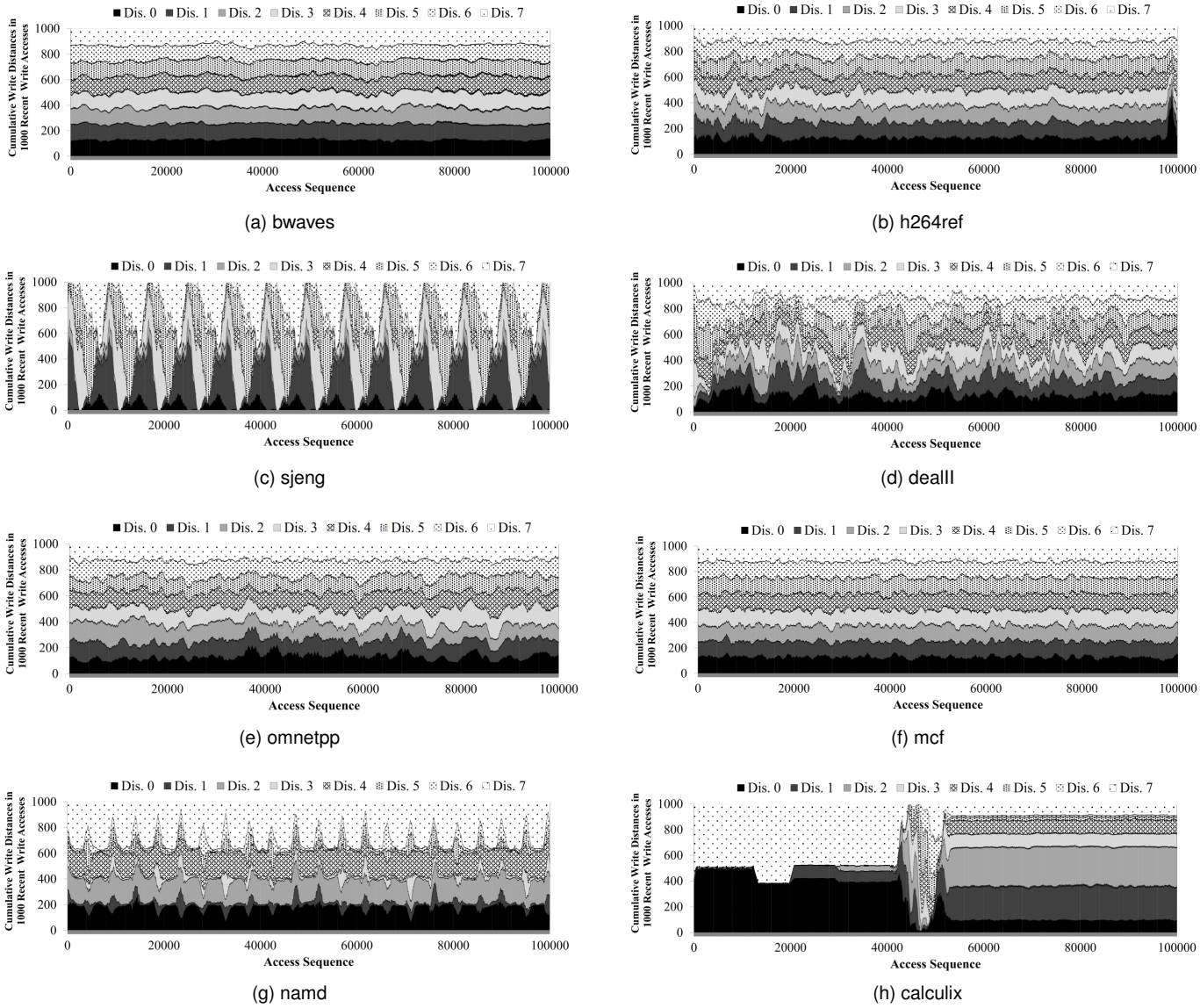
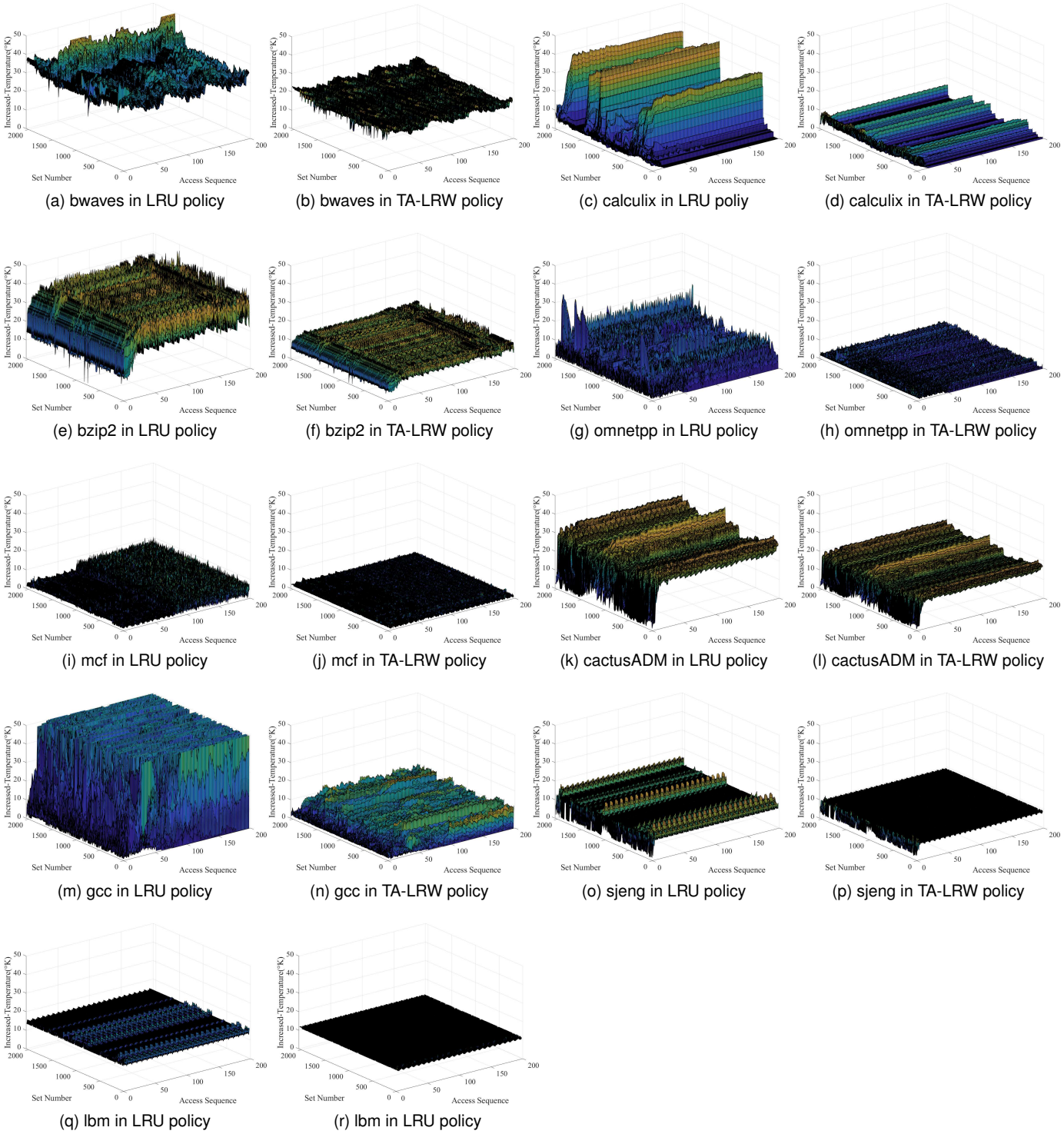


Fig. 16. Accumulated write distances for every 1000 recent writes in (a) bwaves, (b) h264ref, (c) sjeng, (d) dealII, (e) omnetpp, (f) mcf, (g) namd, and (h) calculix workloads in LRU policy.

A large variation is observed in Fig. 17(g) for the increased temperature of LRU policy in *omnetpp* workload, which not only is efficiently smoothed, but also is reduced to less than 5°K by employing TA-LRW policy (Fig. 17(h)).

The temperature increase of LRU policy is not considerable in initial accesses of *mcf* workload, as shown in Fig. 17(i), and is raised to about 10°K afterward. Fig. 17(j) shows that TA-LRW prevents the temperature to increase more than 3°K. Based on Fig. 17(k) and Fig. 17(l), the temperature behavior of *cactusADM* workload in LRU and TA-LRW policies, respectively is similar to that in *bzip2* workload. The largest temperature variation for LRU policy is observed in *gcc* workload (Fig. 17(m)), which indicates 50°K increase for several sets in a considerable number of accesses. Fig. 17(n) depicts that TA-LRW reduces the upper bound of increased temperature to 13°K where the majority values are less than 10°K. Considering *sjeng* workload in Fig. 17(o), the cache temperature is increased by 15°K in LRU policy and TA-LRW reduces this value to about 10°K, as shown in Fig. 17(p). The same behavior is observed for *lbm* workload in LRU and TA-LRW policies, as depicted in Fig. 17(q) and Fig. 17(r), respectively.



(a) bwaves in LRU policy (b) bwaves in TA-LRW policy (c) calculix in LRU poliy (d) calculix in TA-LRW policy

(e) bzip2 in LRU policy (f) bzip2 in TA-LRW policy (g) omnetpp in LRU policy (h) omnetpp in TA-LRW policy

(i) mcf in LRU policy (j) mcf in TA-LRW policy (k) cactusADM in LRU policy (l) cactusADM in TA-LRW policy

(m) gcc in LRU policy (n) gcc in TA-LRW policy (o) sjeng in LRU policy (p) sjeng in TA-LRW policy

(q) lbm in LRU policy (r) lbm in LRU policy

Fig. 17. Temperature increase in blocks of STT-MRAM cache with LRU and TA-LRW policies for (a, b) bwaves, (c, d) bzip2, (e, f) calculix, (g, h) omnetpp, (i, j) mcf, (k, l) cactusADM, (m, n) gcc, (o, p) sjeng, and (q, r) lbm benchmarks.