Work-in-Progress: Enabling Reliable Main Memory Using STT-MRAM via Restore-Aware Memory Management

Armin Haj Aboutalebi
Department of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio, TX 78249
armin.hajaboutalebi@utsa.edu

Lide Duan
Department of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio, TX 78249
lide.duan@utsa.edu

CCS CONCEPTS

• **Computer systems organization** → **Processors and memory architectures**; • **Hardware** → **Memory and dense storage**;

KEYWORDS

STT-MRAM, read disturbance, restore-aware memory management

ACM Reference format:
Armin Haj Aboutalebi and Lide Duan. 2017. Work-in-Progress: Enabling Reliable Main Memory Using STT-MRAM via Restore-Aware Memory Management. In *Proceedings of CASES '17 Companion, Seoul, Republic of Korea, October 15–20, 2017,* 2 pages.
DOI: 10.1145/3125501.3125517

1 ABSTRACT (INTRODUCTION)

As an important non-volatile memory technology, STT-MRAM is widely considered as a universal memory solution in current processors. Employing STT-MRAM as the main memory offers a wide variety of benefits, but also results in unique design challenges. In particular, read disturbance characterizes accidental data corruption in STT-MRAM after it is read, leading to a need of restoring data back to memory after each read operation. These extra restores significantly degrade system performance and energy efficiency, greatly changing the timing scenarios that conventional designs were optimized for. As a result, directly adopting conventional, restore-agnostic memory management techniques may lead to sub-optimal designs for STT-MRAM.

In this work, we propose Restore-Aware Policy Selection (RAPS), a dynamic and hybrid row buffer management scheme that factors in the inevitable data restores in STT-MRAM-based main memory. RAPS monitors the row buffer hit rate at run time, dynamically switching between the open- and close-page policies. By factoring in restores, RAPS accurately captures the optimal design points, achieving optimal policy selections at run time. Our experimental results show that RAPS significantly improves system performance and energy efficiency compared to the conventional policies.

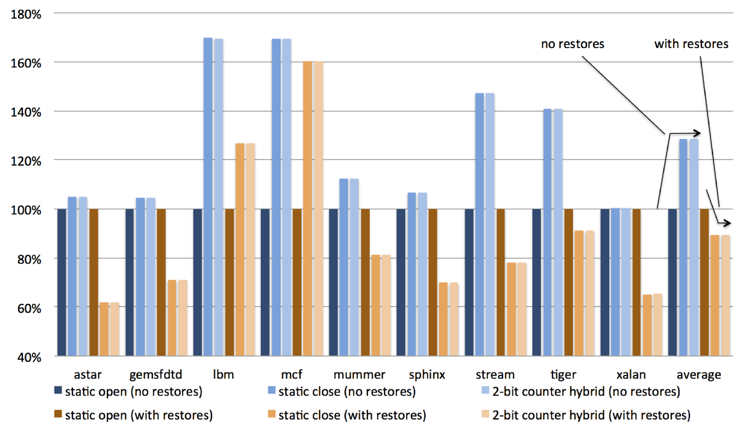

Figure 1: Execution time comparison of three existing page-closure policies when restores are not present (left three bars) and present (right three bars).

2 BACKGROUND

A wide variety of benefits can be obtained in STT-MRAM-based main memories, including the removal of refreshes, nearly zero idle power, almost infinite data retention times, etc. However, employing STT-MRAM as the main memory also results in unique design challenges, such as high write overhead [3] and LPDDR-incompatible sense amplifiers [5]. This paper investigates a critical data reliability challenge, namely **read disturbance** [6] [2], that accidentally corrupts memory data after a read operation to STT-MRAM cells. In order to preserve data integrity under read disturbance, a simple **read-and-restore** scheme [4] has been used.

The conventional row buffer management policy (or page-closure policy) can be open-page or close-page. **The open-page policy** keeps a row open in the row buffer for all ready accesses targeting this row. If a row buffer conflict occurs, it needs to perform a *precharge* and an *activate* before it can read/write the new row. **The close-page policy** opens a row for each read/write operation, and automatically closes it afterwards. It only needs an *activate* when opening a row, but has to do that for every read/write access.

3 MOTIVATION

This work is motivated by the observation that the added data restores in STT-MRAM may significantly change the scenarios that conventional memory controllers were originally optimized for. Since the existing memory management schemes only target

DRAM and are thereby **restore-agnostic**, directly applying them to STT-MRAM may result in suboptimal designs. Figure 1 compares three existing page-closure policies: static open-page, static close-page, and a hybrid policy based on two-bit saturating counters [1]. The two-bit counter hybrid policy uses two-bit saturating counters to characterize the memory access pattern: a row buffer hit/conflict decrements/increments the counter at run time; the counter value is used to predict what policy to use, with values of 0 and 1 indicating open-page and 2 and 3 indicating close-page. First, the open-page policy performs much better than the close-page policy when restores are disabled; when restores are present, the close-page policy becomes much better. This is because the open-page policy is penalized more by restoring the whole row compared to the close-page policy restoring only a cache line of data. Second, the two-bit counter hybrid policy is not effective in making the desired policy selections. This is because it is restore-agnostic, treating the two static policies in a uniform manner, which may not be the optimal choice at run time. Consequently, we need a more realistic, **restore-aware** page-closure policy that can better capture the dynamic memory behavior.

4 DESIGN

In this work, we propose **Restore-Aware Policy Selection (RAPS)**, a dynamic and hybrid row buffer management scheme that factors in the inevitable data restores in STT-MRAM. RAPS keeps track of the row buffer hit rate over a configurable program phase length, determining the desired page-closure policy (open-page or close-page) upon entering a new phase based on the dynamically observed row buffer hit rate.

RAPS relies on an analytical model to determine the row buffer hit rate threshold that distinguishes the two static policies. In open-page, the average read latency can be expressed as:

$$Latency1 = x \cdot tCAS + (1-x) \cdot (tRestorePage + tRP + tRCD + tCAS) \quad (1)$$

where x is the row buffer hit rate; $tCAS$ is the time to access column data; tRP is the precharge time to close the current row; $tRCD$ is the time to activate a new row; and $tRestorePage$ is the time to restore the whole row buffer (page). In open-page, a row buffer hit only incurs $tCAS$; whereas a row buffer conflict involves restoring the current row ($tRestorePage$), precharging the bitlines to close the current row (tRP), activating the new row ($tRCD$), and the column access delay for the requested data ($tCAS$). In close-page, the latency of a read request can be expressed as:

$$Latency2 = tRCD + tCAS + tRestoreLine \quad (2)$$

where $tRCD$ is the time to activate a row; $tCAS$ is the time to access column data; and $tRestoreLine$ is the time to restore a cache line of data. The close-page policy opens a row for every memory access request regardless of row buffer hits/conflicts.

If we compare $Latency1$ and $Latency2$ and solve for x, we have:

$$x = \frac{tRP + tRestorePage - tRestoreLine}{tRP + tRCD + tRestorePage} = th \quad (3)$$

where th is the minimal row buffer hit rate for the open-page policy to perform better than the close-page policy. Therefore, to dynamically achieve better performance, we should choose the open-page policy if the current row buffer hit rate is higher than th, and the close-page policy if the hit rate is lower than th. This threshold value can be pre-calculated at design time using the design parameters shown in Equation 3.

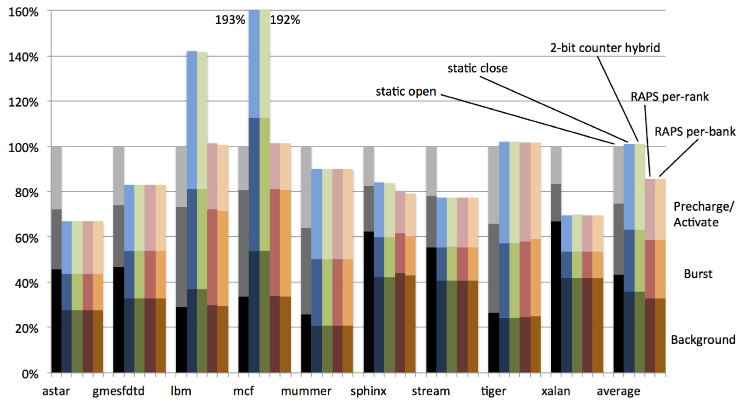

Figure 2: Energy consumption of different schemes.

5 PRELIMINARY EXPERIMENTS

Due to space, we only show the energy results of different policies in Figure 2. All results are normalized to the static open-page policy. *RAPS per-rank* calculates the row buffer hit rate for the entire rank; *RAPS per-bank* monitors the row buffer hit rate and makes policy selection separately for each bank. On average, RAPS achieves a 15% energy reduction compared to the baseline.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a restore-aware page-closure policy selection scheme called RAPS to dynamically select open- or close-page policy based on the row buffer hit rate. Our future work include: (1). validating the benefits of RAPS by evaluating its threshold variation trends and more design configurations; (2). extending RAPS to multiprogrammed and multithreaded workloads; and (3). extending RAPS to time-based page-closure policies.

ACKNOWLEDGMENTS

The work is supported by the National Science Foundation under Grant No. CCF-1566158.

REFERENCES

[1] Mohsen Ghasempour, Aamer Jaleel, Jim Garside, and Mikel Lujan. 2016. HAPPY: Hybrid Address-based Page Policy in DRAMs. In *MEMSYS*.
[2] Lei Jiang, Wujie Wen, Danghui Wang, and Lide Duan. 2016. Improving Read Performance of STT-MRAM based Main Memories through Smash Read and Flexible Read. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*.
[3] E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu. 2013. Evaluating STT-RAM as an energy-efficient main memory alternative. In *ISPASS*.
[4] R. Takemura, T. Kawahara, K. Ono, K. Miura, H. Matsuoka, and H. Ohno. 2010. Highly-scalable disruptive reading scheme for Gb-scale SPRAM and beyond. In *IMW*.
[5] Jue Wang, Xiangyu Dong, and Yuan Xie. 2014. Enabling High-performance LPDDRx-compatible MRAM. In *ISLPED*.
[6] Rujia Wang, Lei Jiang, Youtao Zhang, Linzhang Wang, and Jun Yang. 2015. Selective Restore: An Energy Efficient Read Disturbance Mitigation Scheme for Future STT-MRAM. In *DAC*.