High-Endurance Hybrid Cache Design in CMP Architecture With Cache Partitioning and Access-Aware Policies

Ing-Chao Lin, *Senior Member, IEEE*, and Jeng-Nian Chiou

***Abstract*— In recent years, nonvolatile memory (NVM) technologies, such as spin-transfer torque random-access memory (RAM) (STT-RAM) and phase change RAM, have drawn a lot of attention due to their low leakage and high density. However, both of these NVMs suffer from high write latency and limited endurance problems. To mitigate the write pressure on NVM, many static RAM (SRAM)/NVM hybrid cache designs have been proposed with write management policies. Unfortunately, existing hybrid cache designs do not consider the unbalanced workload of each core in (chip multiprocessor) architecture, resulting in unbalanced wear out of hybrid caches. This paper considers the unbalanced write distribution of a hybrid cache for CMP architecture as well as a novel hybrid cache design that includes SRAM cache, STT-RAM cache, and STT-RAM/SRAM hybrid cache banks. Based on the proposed hybrid cache design, two access-aware policies are proposed to mitigate unbalanced wearout of the STT-RAM region, and a wearout-aware dynamic cache partitioning scheme is proposed to dynamically partition the hybrid cache, improving the unbalanced write pressure among different cache partitions. The experimental results show that our proposed scheme and policies can achieve an average of 89 times improvement in cache lifetime and are able to reduce energy consumption by 58% compared with a SRAM cache.**

***Index Terms*— Dynamic cache partitioning, hybrid bank, hybrid L2 cache, management policy, spin-transfer torque RAM (STT-RAM), wear leveling.**

I. Introduction

IMPROVING processor performance by increasing frequency has become more difficult due to overheating. To overcome this problem, computer architecture has shifted from single-core chip to multicore chip designs, which are also referred to as chip multiprocessors (CMPs). By dividing the workload among different cores, higher performance can be achieved even if the frequency of each core is not increased significantly.

In CMP architecture, the last-level cache (LLC) is typically shared by multiple processors and is composed of static random-access memory (RAM) (SRAM). However, the traditional SRAM cache consumes significant leakage power that dominates the energy consumption of on-chip memory systems. Recently, nonvolatile memory (NVM) technologies, such as phase change RAM (PRAM) and spin-transfer torque RAM (STT-RAM), have drawn a lot of attention due to their low leakage and high density, which make them good candidates for LLC. Table I compares SRAM, STT-RAM, and PRAM. It can be seen that both STT-RAM and PRAM have lower leakage power than SRAM. When comparing STT-RAM with PRAM, STT-RAMs have lower access time, less energy consumption, and higher endurance, which make them a better choice for LLC.

Due to intensive cache writing, hybrid cache architecture that consists of SRAM and STT-RAM was proposed in [16], [22], and [25] to reduce the high write latency and high write energy of STT-RAM, as shown in Fig. 1. Most of the LLCs are composed of STT-RAM, and only some of them are SRAM. The STT-RAM bank that is closest to a core is the local bank of the core, and the core has minimum access latency to the banks. With the help of a small SRAM region and appropriate write management, the write pressure on the STT-RAM region can be reduced, and the average

TABLE I
COMPARISON BETWEEN SRAM AND NVMS [6], [7]

	SRAM	STT-RAM	PRAM
Density	1X	4X	16X
Read time	Very fast	Fast	Slow
Write time	Very fast	Slow	Very slow
Read energy	Low	Low	Medium
Write energy	Low	High	High
Leakage energy	High	Low	Low
Endurance	10^{16}	4×10^{12}	10^{9}

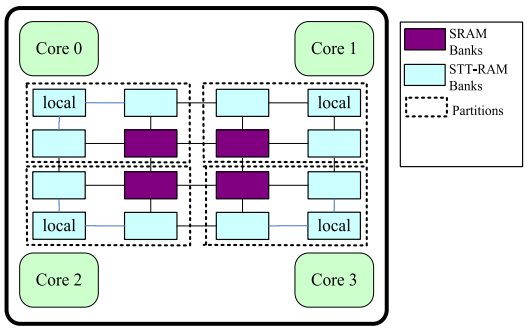

Fig. 1. Hybrid cache for CMP architecture.

Manuscript received October 12, 2013; revised July 14, 2014; accepted August 18, 2014. Date of publication October 31, 2014; date of current version September 23, 2015. This work was supported in part by the National Science Council of Taiwan under Grant NSC-102-2221-E-006-281 and in part by the Ministry of Science and Technology, Taiwan, under Grant 103-2221-E-006-255.

The authors are with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan (e-mail: iclin@mail.ncku.edu.tw; p76011103@mail.ncku.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2014.2361150

TABLE II
SYSTEM CONFIGURATION USED IN SIMULATION

Processor	4 cores, 2 GHz
L1 I/D cache	32/64 KB per core, 4-way, 64B block size
Shared L2 cache	16 MB(1MB x 16 banks), 16-way, 64B block size

access latency can be improved. The energy consumption can be reduced as well as compared with a SRAM-only cache.

However, the STT-RAM write endurance issue, which has often been ignored by earlier works, needs to be considered. Currently, STT-RAM write cycles are limited by about 4×10^{12} [6], as shown in Table I, and reliable writes are not guaranteed if write cycles are higher. It has been reported that when a medical image application, *segmentation*, is executed on a 4 GHz CPU with 32 kB L1 cache and 2 MB STT-RAM L2 cache continuously [3], the lifetime of a STT-RAM is only 2.17 years without any optimization applied. This issue is even more serious on multilevel cell STT-RAM since its write current is higher single-level cell STT-RAM [6], [14].

Another issue with CMP LLCs is the interference among competing processes that leads to increased shared resource contention and conflict misses in LLCs. One solution to mitigate this problem is to partition the shared cache into several private partitions for each core, thus reducing cache contention. Furthermore, to improve cache utilization, dynamic cache partitioning schemes have been proposed to manage resources through allocating specific portions of the caches to each core according to their workload profiling information [20], [21].

Current related dynamic cache partitioning works focus on increasing CMP performance and the private partition size for a core is determined by the cache miss rate and instructions per cycle. However, when using STT-RAM as the LLC, cache partitioning should also consider the STT-RAM write endurance problem. Otherwise, unbalanced write traffic to each partition will reduce the lifetime of the STT-RAM cache.

To demonstrate the unbalanced write distribution, we simulate a four-core CMP machine using a modified gem5 simulator [1]. The benchmarks are selected from PARSEC 2.1 [3] and the details of the system configuration are listed in Table II. The L2 cache is dynamically partitioned according to the workloads, while the write traffic to each bank is not considered. The simulation results show the write accesses to each bank to be nonuniformly distributed over the sixteen banks in all benchmarks, and the four most unbalanced cases are shown in Fig. 2. As a result, the cache banks that suffer from large write accesses will wear out faster than other banks.

Therefore, it is critical to consider the write pressure to each bank during cache partitioning. However, current cache partitioning techniques are not suitable for hybrid caches since the unbalanced write traffic among different banks is not considered. In addition, local banks suffer from more serious lifetime problems since they normally receive more write pressure than other cache banks. To solve these problems, we propose a novel hybrid cache architecture that contains

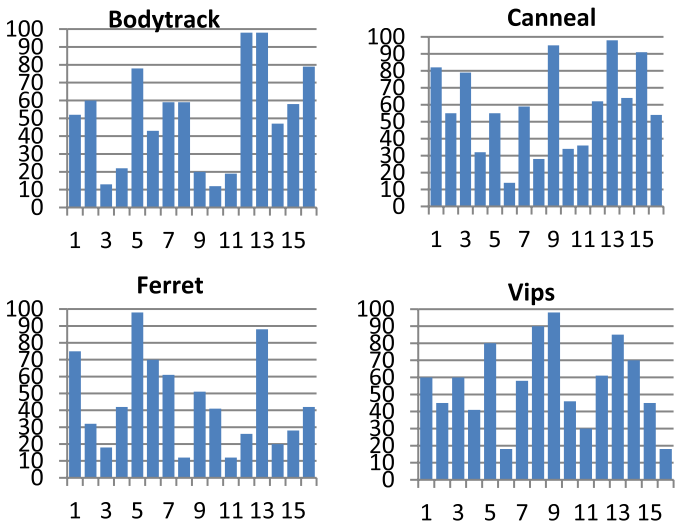

Fig. 2. Nonuniformity of writes among the sixteen banks.

three types of cache banks: SRAM banks, STT-RAM banks, and STT-RAM/SRAM hybrid banks. In the meantime, two access-aware policies and a wearout-aware dynamic cache partitioning scheme are proposed. The contributions of this paper can be summarized as follows.

1) *A Novel Hybrid Cache Architecture That Contains SRAM Banks, STT-RAM Banks, and STT-RAM/SRAM Hybrid Banks for Chip Multiprocessors:* To reduce write pressure on the local bank of a core, we propose a new hybrid bank architecture that combines both STT-RAM and SRAM. With the help of the SRAM region in the local bank, the write pressure on the STT-RAM region can be migrated to the SRAM region in the local bank and can be significantly reduced.
2) *Access-Aware Policies to Manage Hybrid Cache:* To reduce the write traffic to the STT-RAM regions and balance the write distribution over the STT-RAM regions, we propose an STT-RAM write management policy. In addition, to improve the SRAM write utilization, a SRAM read management policy that moves less-written data from SRAM to STT-RAM is proposed.
3) *Wearout-Aware Dynamic Cache Partitioning to Improve Lifetime:* To make the write pressure uniform among different cache partitions, we propose a wearout-aware dynamic cache partitioning scheme. The scheme considers the different workload write-pressure from each core to change the capacity of SRAM and STT-RAM regions in each partition. Therefore, the write pressure among cache partitions is balanced.
4) *Sensitivity Analysis of Different Ratio of SRAM and STT-RAM:* To evaluate the proposed hybrid cache design, we analyze the unbalanced write distribution with the proposed hybrid banks under different SRAM and STT-RAM ratios. The sensitivity analysis provides comprehensive results of the effectiveness of the proposed technique under different SRAM and STT-RAM size ratios.
5) The experimental results show that the proposed hybrid cache with access-aware policies and a wearout-aware

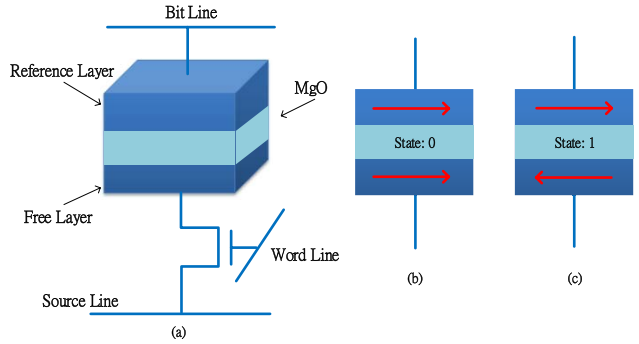

Fig. 3. Overview of STT-RAM cell. (a) Schematic of one STT-RAM cell. (b) Parallel low resistance, representing 0 state. (c) Antiparallel high resistance, representing 1 state.

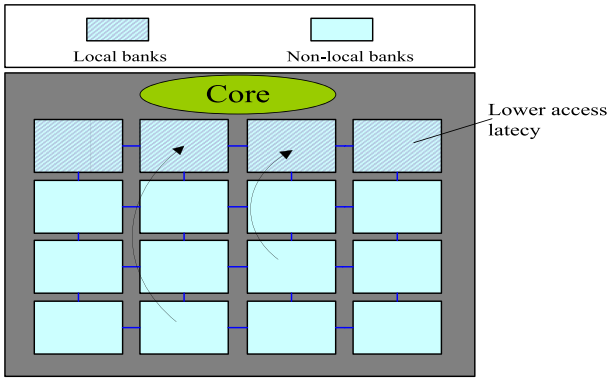

Fig. 4. D-NUCA architecture. Nonlocal data are moved to local banks to reduce access latency.

dynamic cache partitioning scheme can improve the lifetime of a hybrid cache by 89 times on average. In addition, 58% of the total energy consumption can be reduced as compared with a SRAM cache.

The rest of this paper is organized as follows. Section II introduces the fundamentals of STT-RAM and related work for hybrid caches. Section III presents the proposed methodology, including the proposed hybrid cache architecture, access-aware policies, and the wearout-aware dynamic cache partitioning scheme, and Section IV details the experimental results. Section V concludes this paper.

II. Preliminaries

This section first introduces the fundamentals of an STT-RAM cell and then describes the related works on nonuniform and hybrid L2 cache designs.

A. STT-RAM Fundamentals

The STT-RAM discussed in this paper is a new generation of magnetoresistive random-access memory (MRAM). As shown in Fig. 3, STT-RAM uses a magnetic tunnel junction (MTJ) as the memory storage instead of electric charges. An MTJ consists of two ferromagnetic layers and one tunnel barrier layer. One of the ferromagnetic layers is called a reference layer and it has a fixed magnetic direction. The other ferromagnetic layer is called a free layer whose magnetic direction can be changed through a switching current. When the two magnetic directions of the reference and free layers are parallel, the resistance of the MTJ is low, representing a 0 state. Conversely, when the two layers have different magnetic directions (antiparallel), the MTJ resistance is high, representing a 1 state. A low current is used to read the MTJ state, while a high current of an appropriate polarity is used to change the magnetic state that involves long write latency and high write energy consumption.

B. Related Works for Nonuniform and Hybrid Cache Designs

With the increase in cache capacity and area, modern L2 cache memory is subdivided into multiple banks. Without considering the different memory technologies of each bank, the local bank that is closer to the processor has shorter access latency due to shorter wire delay. This architecture is called nonuniform cache access (NUCA) [15]. In an NUCA-based cache, the application access patterns to local and remote banks should be considered since accessing the local banks is faster than accessing the remote banks. One common way to improve system performance is to dynamically move frequently accessed data during runtime to local banks. This technique is called dynamic NUCA (D-NUCA) [15]. Fig. 4 shows a simple L2 cache with D-NUCA architecture. The L2 cache consists of sixteen banks. Frequently accessed data are moved to local banks to reduce access time.

In recent years, hybrid cache architecture has attracted considerable attention. The concept of integrating NVMs with SRAM to construct hybrid caches was previously proposed in [5], [16], [18], [22], and [25]. Sun *et al.* [22] first stacked an MRAM-based L2 cache directly atop CMPs and proposed two architectural techniques: a read-preemptive write buffer and a SRAM/STT-RAM hybrid L2 cache to improve the performance reduction problem due to long NVM write latency. Wu *et al.* [25] proposed a read–write-aware hybrid cache architecture design, and the cache was partitioned into read and write regions comprised of STT-RAM and SRAM, respectively, to improve cache performance. Li *et al.* [16] integrated SRAM with STT-RAM to construct a novel hybrid cache architecture for CMPs, and they also proposed energy-aware read and write mechanisms for the hybrid cache to improve performance for workloads with different write patterns. Lee *et al.* [18] considered the L1 SRAM cache and external memory to analyze hybrid cache architecture, and models for average memory access time, power consumption, and area of cache memory were proposed to compare various cases adopting different memory types and benchmark programs. However, these works did not consider the endurance issues of NVMs due to unbalanced write distribution. Jadidi *et al.* [13] pointed out that the nonuniformity of writes in STT-RAM cache lines results in faster wearout of some lines as compared with others, and a remapping method was proposed to balance the write accesses among each STT-RAM cache sets and lines to extend cache lifetime.

There have also been related works investigating traditional cache issues for hybrid cache architecture [7], [11], [19], [23].

Ahn *et al.* [2] proposed a method for hybrid cache architecture that can predict and bypass dead writes to reduce energy consumption. Wang *et al.* [24] proposed a block placement and migration to improve performance and reduce energy. Chen *et al.* [7] proposed a reconfigurable hybrid cache, which can be reconfigured as powered ON or OFF for the SRAM or NVM arrays in L2 cache architecture. A power gating circuit allows the hybrid cache to dynamically power OFF the cache ways based on the cache workload to save energy while maintaining system performance. Hsieh and Kuan [11] indicated that the least recently used (LRU) replacement algorithm, which works well for traditional dynamic RAM (DRAM) caches, should not be directly applied to DRAM/PRAM hybrid caches because of the endurance issue in PRAM. They proposed a double circular caching scheme to manage a hybrid cache through the migration of the cached data between a DRAM cache and a PRAM cache. Mao *et al.* [19] determined that the additional cache write operations induced by data prefetching on multicore systems can increase the write pressure on hybrid caches. To reduce the negative performance impact of aggressive prefetching, they proposed two priority techniques to different types of LLC requests and dynamically identified an LLC nonintensive application. Wang *et al.* [23] indicated that NVM technologies usually have limited write endurance and the process variation can cause some cells to wear out much faster than others. They presented a hard failure-tolerant architecture for nonvolatile caches, improving cache lifetime under different process variations. To the best of our knowledge, this paper is the first work to investigate dynamic cache partitioning and unbalanced write pressure among different partitions for a hybrid cache in CMP architecture.

III. Proposed Methodology

This section begins with the introduction of the local hybrid bank architecture, which is proposed to balance the write distribution among local and nonlocal banks. Then, access-aware policies, which include STT-RAM write management policy and SRAM read management policy, are proposed to reduce the write pressure and balance the write distribution on STT-RAM regions. Finally, since the workloads for each core may be unbalanced in multicore architecture, a dynamic cache partitioning scheme is proposed to partition the hybrid L2 cache dynamically to balance the write pressure in each partition.

A. Local Hybrid Bank in Hybrid Cache Architecture

The hybrid cache design used in this paper is based on D-NUCA for better performance, as shown in Fig. 5(a). The local banks have lower access latency and higher access frequency than other cache banks and the STT-RAM occupies a larger area than the SRAM to increase capacity and reduce the total energy consumption of the hybrid cache. Compared with SRAM, STT-RAM has lower leakage and higher density. Thus, in a hybrid cache, the capacity of the STT-RAM regions is much larger than that of the SRAM regions. Since the write operation of SRAM consumes less energy and access time

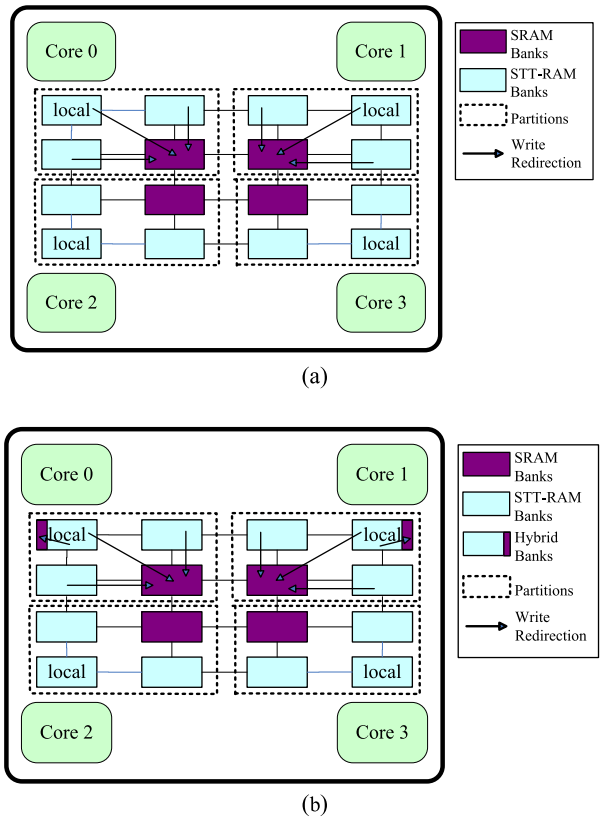

Fig. 5. Overview of data migration and write redirection in a hybrid L2 cache. (a) Write redirection flow caused by write management policy. (b) Write redirection flow when applying a hybrid bank and a write management policy.

than STT-RAM, Fig. 5(a) also shows that if we appropriately redirect the write operation from STT-RAM to SRAM, the hybrid cache can achieve better performance and reduce energy consumption. The redirection can be controlled through the cache controller with a write management policy that will be introduced in the following sections.

Reducing the write pressure on STT-RAM also extends the shorter lifetime of STT-RAM as compared with SRAM. However, when an STT-RAM intends to redirect its write traffic to a SRAM cell, there is high chance that the SRAM cell is already occupied by the write traffic from another STT-RAM cell and that the redirection will fail. This issue is more serious in the case of the local bank since the bank has less latency and normally receives a significant amount of write traffic from its core in the D-NUCA architecture. Consequently, local banks will suffer from larger write pressure and wear out faster than nonlocal banks.

To relieve the large pressure on local banks, we propose to integrate STT-RAM with SRAM to construct a local hybrid bank. The SRAM region in a local hybrid bank is designed to receive the write traffic to the local hybrid bank. To prevent the SRAM in the hybrid bank from being occupied by the write traffic of other banks, only the write traffic to the local hybrid bank can be redirected to the SRAM region in the local hybrid bank. Fig. 5(b) shows the write redirection flow in a hybrid cache when applying a hybrid bank and write management policy. When the write operations in a hybrid

Algorithm 1 STT-RAM Write Management Policy

```
1.  if a write request hits on a STT-RAM line, line {
2.    if there are invalid SRAM lines
3.      Redirect the write request to invalid SRAM lines
4.      Invalidate line
5.    else if there are another invalid STT-RAM lines
6.      Redirect the write request to other STT-RAM lines
7.      Invalidate line
8.    else
9.      Perform the write operation on line
10. }
11. else if a write miss on a STT-RAM line{
12.   if SRAM has invalid lines
13.     Redirect the write request to invalid SRAM lines
14.   else
15.     Select LRU STT-RAM line to be the victim
16.     Perform the write operation on the victim
17. }
```

Algorithm 2 SRAM Read Management Policy

```
1.  if a read request hits on a SRAM line, line {
2.    if line is in LRWS and all SRAM lines are in use {
3.      if there are invalid STT-RAM lines
4.        Copy the data from line to invalid STT-RAM line
5.        Redirect the read operation on invalid STT-RAM line
6.        Invalidate line
7.      else
8.        Select LRU STT-RAM line to be the victim
9.        Copy line to the victim
10.       Redirect the read operations on the victim
11.       Invalidate line
12.   }
13.   else if line is not is LRWS{
14.     Perform the read operation on line
15.   }
16. }
```

bank need to be redirected from STT-RAM to SRAM to reduce write pressure, they will be first redirected to the SRAM region in the hybrid bank. If the entire SRAM region in a local hybrid bank is in use, then the hybrid bank will redirect the write traffic to the SRAM banks. As a result, the unbalanced write pressure between the local and nonlocal banks can be improved. Note that only the STT-RAM banks and the STT-RAM region in hybrid banks have the issue of write pressure, while the SRAM region does not need to consider write pressure.

B. Access-Aware Policies

To reduce the write pressure on STT-RAM and increase the write utilization of SRAM, two access-aware policies are proposed to distribute the write traffic to the hybrid cache: a STT-RAM write management policy and a SRAM read management policy.

1) STT-RAM Write Management Policy: The main ideas of our proposed STT-RAM write management policy are: 1) to reduce the write traffic to the STT-RAM and 2) to make the write distribution uniform over the STT-RAM. Algorithm 1 shows the details for the STT-RAM write management policy. When a write request has the cache hit of an STT-RAM line and there is an invalid SRAM line, the write request will be redirected to the invalid SRAM line and the STT-RAM line will be invalidated (lines 2 to 4). Note that the SRAM lines in the hybrid bank can only receive the write request from the STT-RAM line of the same hybrid bank. Next, if there is no invalid SRAM line, then the data will be written into STT-RAM lines. To avoid a large amount of write requests on the same STT-RAM line, the write request should be redirected to another invalid STT-RAM line instead of being redirected to the original cache line (lines 5 to 7). If there are no other invalid STT-RAM lines, the write request will be performed on the original cache line (lines 8 to 9). Lines 11 to 16 describe the details when a write miss occurs in the STT-RAM lines. When a cache miss in the STT-RAM lines occurs, and there is an invalid SRAM line, the write request is redirected to the SRAM lines (lines 11 to 13); otherwise, an STT-RAM line is selected as a victim based on LRU policy and must be written back to the lower level memory for a write-back cache. Then, the write operation is performed on the selected victim (lines 14 to 16).

2) SRAM Read Management Policy: To further increase the write utilization of SRAM, the less-written data that occupies the SRAM lines should be removed. When this less-written data is removed from SRAM, more space is left for write operation. Based on the LRU policy, we define a less recently written set (LRWS) which has n SRAM lines. The set contains the least recently written (LRW_1), the second least recently written (LRW_2), and the third least recently written (LRW_3), up to the nth least recently written (LRW_n) cache lines in SRAM. All the cache lines in the LRWSs are considered to be the candidates that should be moved to STT-RAM. In our paper, n is set to 10% of the SRAM capacity in each partition.

Algorithm 2 shows the details of the SRAM read management policy. When a read request has the cache hit of the SRAM line in a LRWS, and all other SRAM lines are in use, the read request will be redirected to other cache lines (lines 1 to 11). If there are invalid STT-RAM lines, the read request will be redirected to the invalid STT-RAM line, and the SRAM line will be invalidated (lines 3 to 6). Otherwise, the LRU STT-RAM line is selected to be the victim and the read request is redirected to the victim. Then, the SRAM line is invalidated (lines 8 to 11). Although moving the less-written data from SRAM to STT-RAM incurs additional writes on STT-RAM, the increased write utilization of SRAM can reduce the total write traffic to the STT-RAM. If the SRAM line is not in the LRWS or all the other SRAM lines are not in use, the read request will be performed on the original cache line (lines 13 to 15).

3) Address Remapping: Our proposed policies have three types of redirection for cache requests. The first is the redirection from STT-RAM to SRAM (Algorithm 1, line 3 and line 13); the second is the redirection from SRAM to STT-RAM (Algorithm 2, line 5 and line 10), and the third is the redirection from STT-RAM to STT-RAM (Algorithm 1, line 6). To implement cache request redirection,

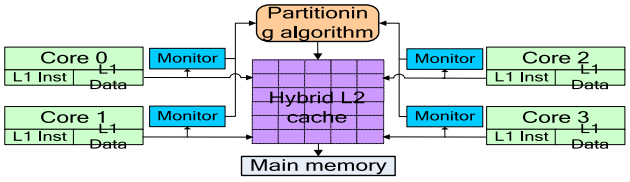

Fig. 6. Framework for dynamic cache partitioning.

there are normally two approaches: *table-based address remapping* and *algebraic address remapping*. A table-based address remapping approach which incurs significant storage overhead keeps track of the original and redirected addresses through a table. In addition, table look-up requires additional energy and latency to each access and scales linearly with the capacity of a hybrid cache. The overhead related to tracking addresses can be avoided by using algebraic address remapping. This method uses an algebraic function to calculate the remapped address instead of looking up the table, and is more practical compared with a table-based approach. Hence, this paper uses the algebraic address remapping approach proposed in [26], which is referred to as *RanCAR*, to implement the redirection of cache requests.

C. Dynamic Cache Partitioning Scheme

When a hybrid cache is used in multicore architecture, an appropriate dynamic cache partitioning technique is needed. Different from a traditional cache, a partitioning hybrid cache should consider unbalanced write traffic to each partition to improve the endurance of STT-RAM. In this section, we propose a dynamic cache partitioning scheme that considers the different workloads in each core to change the capacity of the SRAM and STT-RAM regions in each partition. Fig. 6 shows our framework. Each core has its own private L1 instruction and data cache. The hybrid L2 cache accesses from each core are monitored by their corresponding monitors. Then, with the collected information from each monitor, the proposed partitioning algorithm decides how to allocate the shared cache to each partition. The proposed partitioning algorithm is associativity based cache partitioning, which allocates a fixed number of ways to each partition, and all ways in the hybrid L2 cache will be assigned to a core. A partition example for hybrid cache is shown in Fig. 1. In this example, the L2 cache is divided into four equal partitions, and the private partition of each core consists of a SRAM bank and three STT-RAM banks.

The partitioning algorithm partitions a hybrid cache based on the following criteria: 1) which partition's size needs to be changed and 2) which region in a partition needs to be changed (SRAM or STT-RAM). To determine how to better partition a hybrid cache, we monitor the write counts on SRAM and STT-RAM for each core over time and maintain the number of SRAM and STT-RAM ways owned by each core. With the information of write counts and way number for each partition, two variables are defined as follows:

$$WPSW = \frac{\text{write counts in SRAM}}{\text{number of SRAM ways}}$$
$$WPNW = \frac{\text{write counts in STT-RAM}}{\text{number of STT-RAM ways}}.$$

Algorithm 3 Cache Partitioning Algorithm

```
1.  For each period {
2.    W_SRAM = {WPSW_0, WPSW_1, ..., WPSW_n}
3.    W_STT-RAM = {WPNW_0, WPNW_1, ..., WPNW_n}
4.    Sort array W_SRAM by non-increasing order
5.    Sort array W_STT-RAM by non-increasing order
6.    Select the partition that has largest WPNW value, partition_i
7.    Select the partition that has smallest WPSW value,
      partition_j
8.    if (i != j)
9.       decreases a SRAM way in partition_j
10.      increases a SRAM way partition_i
11.   else
12.      Select partition that has smallest WPNW value,
      partition_k
13.      partition_k decreases a STT-RAM way
14.      partition_i increases a STT-RAM way
15.   }
```

The variables *WPSW* and *WPNW* are the write counts per SRAM way and the write counts per NVM way, respectively. The larger *WPSW* value of a partition means the write utilization of the SRAM in the partition is higher. Similarly, if a partition has a larger *WPNW* value, then the write utilization of STT-RAM in that partition will be higher, leading to faster wear out than other partitions with smaller *WPNW* values. By detecting the write utilization for SRAM and STT-RAM in each cache partition, appropriate cache ways are allocated to the partition that has large write pressure in its STT-RAM region. Therefore, the unbalanced write pressure among cache partitions is improved. Pseudocodes for our partitioning algorithm are shown in Algorithm 3.

At each time interval period, the values of *WPSW* and *WPNW* for each partition are computed. Then, two arrays (W_{SRAM} and $W_{\text{STT-RAM}}$) are used to store their values and are sorted in nonincreasing order (lines 1 to 5). Next, we choose the partition that has the most write count per STT-RAM way (line 6), $partition_i$, and the partition that has the least write count per SRAM way, $partition_j$ (line 7).

When i is not equal to j, a SRAM way from the partition that has the smallest *WPSW* value will be allocated to the partition that has the largest *WPNW* value if both partitions do not belong to the same core (lines 8 to 11). This is because a SRAM way can mitigate more write pressure than a STT-RAM way. Therefore, allocating a SRAM way to a partition can reduce more *WPNW* values than allocating a STT-RAM way.

When i is equal to j, an STT-RAM way from the partition that has the smallest *WPNW* will be allocated to the partition that has the largest *WPNW* value (lines 12 to 15). This is because partition$_i$ also has the smallest *WPSW* value, which means it has low write utilization for SRAM and increasing SRAM ways for this partition will not significantly mitigate the write pressure on STT-RAM ways effectively. Therefore, instead of increasing SRAM ways, STT-RAM ways should be increased in $partition_i$. An STT-RAM way from the partition that has the smallest WPNW, $partition_k$, is allocated to $partition_i$ (lines 12 to 14). Using the proposed cache

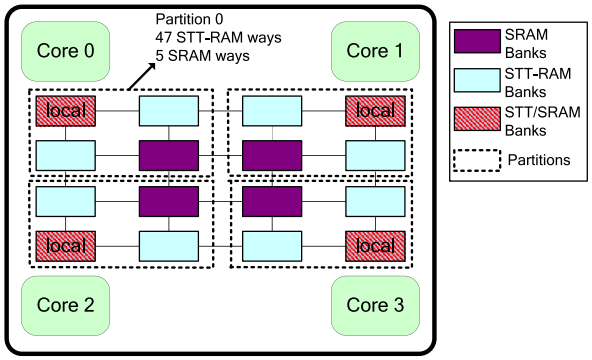

Fig. 7. CMP hybrid cache architecture with equal partition sizes.

partitioning algorithm, the write pressure among the cache partitions can be balanced.

IV. Experimental Setup and Results

Fig. 7 shows our proposed architecture, which includes a four-core CMP system and a shared hybrid L2 cache. The private L1 cache in each core is not shown. The hybrid cache consists of four SRAM cache banks, eight STT-RAM cache banks, and four STT-RAM/SRAM hybrid cache banks, which are connected through an interconnect network. The capacity density of the STT-RAM cache is about four times higher than that of the SRAM cache [22]. To achieve a similar area for each bank, the cache capacity of each STT-RAM and SRAM bank used in our work were set at 2 MB (16-way) and 512 kB (4-way), respectively. In addition, the proposed hybrid bank contains 1920-kB STT-RAM and 128-kB SRAM, which provides a total of 2 MB of cache (15-way + 1-way) with a slightly larger area. Another way to see the shared L2 cache is as a 26-MB hybrid cache that consists of a 208-way equivalent STT-RAM and SRAM ways. Note that SRAM banks are used in the middle of the hybrid cache instead of local banks. The main reason is that the proposed partitioning algorithm can assign different numbers of SRAM banks to each core based on the write pressure of each core; therefore, SRAM banks can be used more effectively.

We use a gem5 Simulator to simulate a 2-GHz processor with four cores [1]. The classic memory model in the gem5 is modified to implement the hybrid L2 cache with nonuniform access times, the access-aware policies, and the dynamic cache partitioning scheme. Table III details the main simulation parameters applied in our experiments. We use the PARSEC 2.1 benchmark suite [3] as the parallel workload to evaluate the proposed scheme. To obtain a reasonable evaluation, all benchmarks are fast forwarded to the beginning region of interest, and then the next 100 M instructions are considered to be a warm up period. The latency and energy values of the SRAM cache used in this paper are obtained from CACTI [28], and for the STT-RAM cache, we use a modified NVSim [8] and the model reported in [22] to obtain the parameters in 65-nm technology, which are given in Table IV.

A. Write Pressure on Local Banks

We first demonstrate that local banks have higher write pressure. To achieve this, we simulate all benchmarks and calculate the write requests counts on each STT-RAM bank and the average standard deviation of write distribution. The write requests to SRAM banks are ignored.

It can be seen that write requests are not evenly distributed to the STT-RAM banks and the standard deviation of write distribution is about 0.097 on average (Fig. 8). The local banks suffer nearly twice as much write pressure compared with other nonlocal banks due to the high access frequency of local banks. Hence, the serious write pressure on local banks needs to be improved.

Fig. 9 compares the local bank lifetime between hybrid local banks and STT-RAM local banks. It can be observed that the

TABLE III
System Configuration Used in Simulation

Processor	4 cores, 2 GHz
L1 I/D Cache	32/64 KB per core (private), 4-way, 2cycles latency, 64B block size, write back
Hybrid L2 Cache	8 STT-RAM banks, 4 SRAM banks, 4 STT-RAM/SRAM hybrid banks, write back
SRAM bank	512KB, 4-way set associative, 64B block size
STT-RAM bank	2MB, 16-way set associative., 64B block size
STT-RAM/SRAM hybrid bank	2MB(1920KB+128KB), 16-way(15+1) set associative, 64B block size
L2 Cache Access Lat.	SRAM: 6 cycles STT-RAM read/write: 6/23 cycles
Coherence Mechanism	MESI directory protocol
Network	Ring network, one router per bank, 3 cycles router and 1 cycle link latency
Repartitioning Period	100M cycles
Main memory	512MB, 60 cycles latency

TABLE IV
Timing and Latency Parameters of SRAM, STT-RAM, and STT/SRAM Cache Banks

Bank type	Tech.	Op.	Lat.(ns)	Energy(nJ)	Leak.(W)
SRAM bank (512KB)	SRAM	R/W	1.74	0.609	0.3691
STT-RAM Bank (2MB)	STT-RAM	R	2.99	0.598	0.0686
		W	11.41	4.375	
STT/SRAM Hybrid bank (1920/128KB)	SRAM	R/W	1.57	0.219	0.0764
	STT-RAM	R	1.71	0.569	
		W	11.32	4.215	

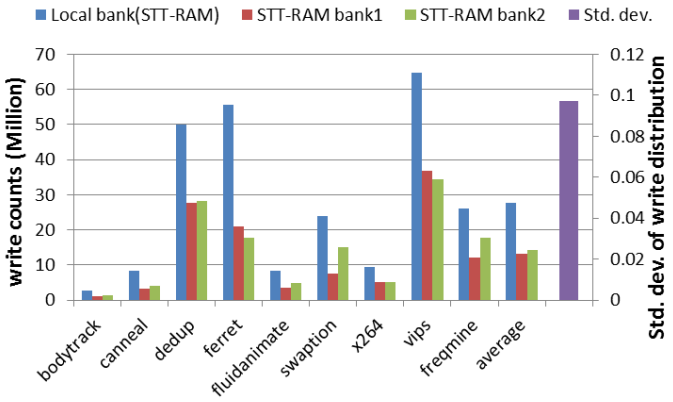

Fig. 8. Unbalanced write distribution on different banks (local banks suffer the highest write pressure.

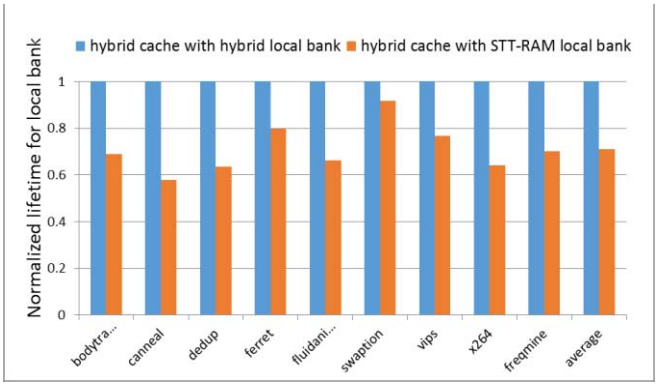

Fig. 9. Local bank lifetime comparison between hybrid local banks and the STT-RAM local banks.

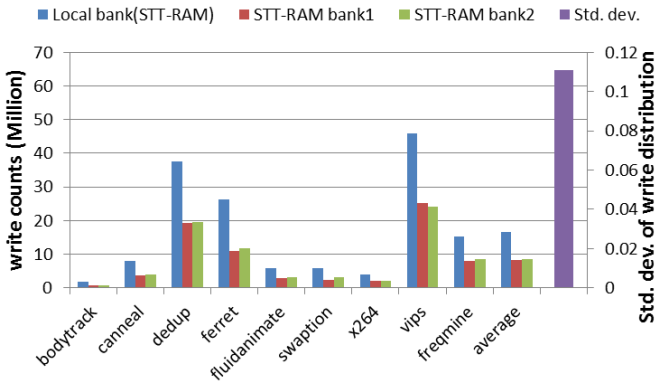

Fig. 10. Unbalanced write count distribution on different banks with access-aware policies and an equal-partition scheme (local banks suffer the highest write pressure).

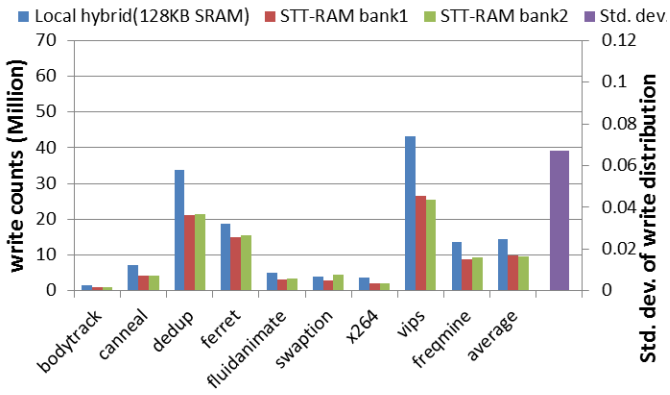

Fig. 11. Improved write counts distribution on local and nonlocal banks with a hybrid bank, access-aware policies, and an equal-partition scheme.

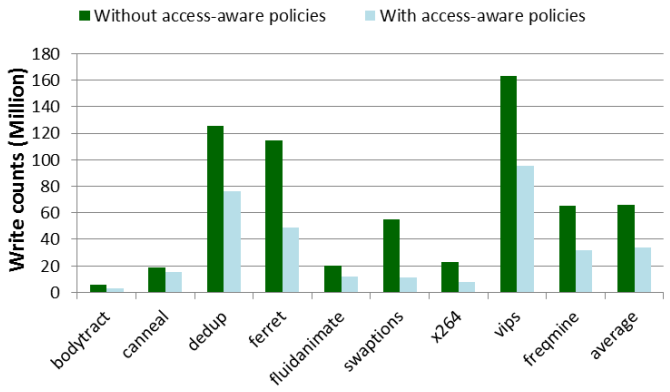

Fig. 12. Comparison of write counts on STT-RAM regions with and without the proposed access-aware policy.

lifetime of hybrid local banks is 24.5% higher than that of the STT-RAM-based local banks. This is because the write pressure of STT-RAM regions in hybrid local banks can be redirected to SRAM regions in hybrid local banks.

B. Write Pressure Comparsion With and Without Hybrid Banks

This section demonstrates that the proposed local hybrid banks can effectively reduce write pressure on local banks. The following experiments are done on a baseline cache scheme called *equal-partition*, which assigns equal sizes to each cache partition, as shown in Fig. 1.

Figs. 10 and 11 compare write count distributions with and without using hybrid banks. The local banks in Fig. 10 are STT-RAM banks only, but the local banks in Fig. 11 are hybrid banks. Note that access-aware policies are applied in both cases. This is because to enable the functionality of hybrid banks, access-aware policies are required. The write counts in Figs. 10 and 11 are lower than those shown in Fig. 7, because access-aware policies redirect the write requests from STT-RAM to the SRAM region.

It can be observed from Fig. 10 that the write pressure on the local banks is still twice as large as that of nonlocal banks and the standard deviation of write distribution is about 0.11 on average. However, when hybrid banks are used, the standard deviation of write distribution is decreased to 0.067 on average, as shown in Fig. 11. With the small SRAM in the proposed hybrid bank, the higher write pressure on local banks can be significantly reduced, improving the unbalanced hybrid cache write distribution.

C. Write Pressure and Average Standard Deviation Reduction Using Access-Aware Policies

In this section, we compare the results with and without our proposed access-aware policies in the equal-partition scheme. Fig. 12 compares the write counts on the STT-RAM region with and without the access-aware policies. It can be seen that the write counts to STT-RAM for the proposed access-aware policies are lower than those without access-aware policies. On average, about 43% of the writes counts on the STT-RAM region are reduced compared with that without access-aware policies. Therefore, the proposed access-aware policies can effectively redirect write counts from STT-RAM to the SRAM region.

Then, the average standard deviation of write counts on the STT-RAM is computed, which represents the nonuniform degree of write distribution. Fig. 13 compares the standard deviation of the write counts. It can be observed that there is about an 84% reduction in the standard deviation of write counts on the STT-RAM on average,

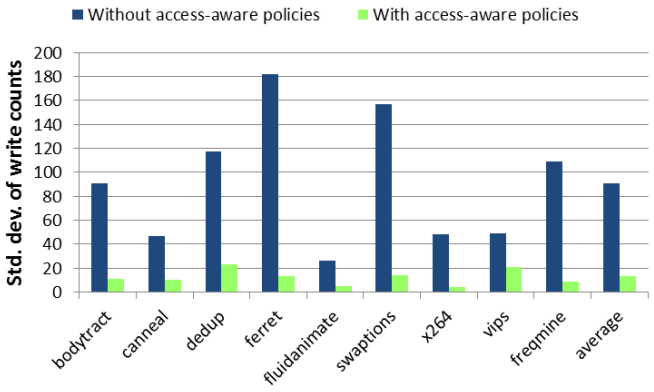

Fig. 13. Comparison of standard deviation of write counts on STT-RAM regions with and without access-aware policy.

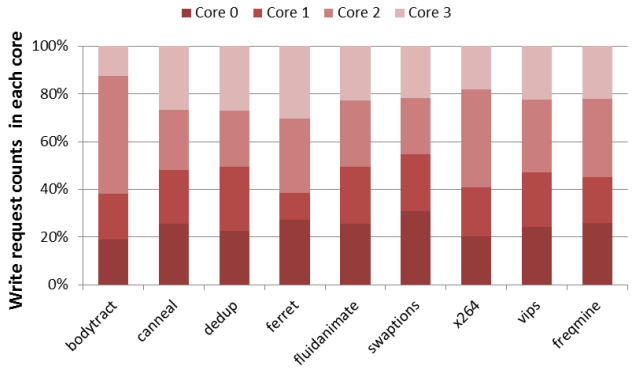

Fig. 14. Unbalanced write counts distribution among four cores in CMPs.

which means the nonuniform distribution of writes over STT-RAM cells is highly improved. From Figs. 12 and 13, it can be observed that our access-aware hybrid cache management policies can balance the write distribution over STT-RAM cells in all benchmarks and significantly reduce write counts to STT-RAM.

D. Unbalanced Write Distribution Improvement Using Dynamic Cache Partitioning Scheme

Fig. 14 shows the percentage of the L2 cache write requests issued from each core. It can be observed that write requests are not balanced, which is a strong indication of the need to change the cache partition size according to the write counts to improve STT-RAM endurance. Then, we compare the percentage of write counts to each cache bank between using the equal-partition scheme and using the proposed dynamic cache partitioning scheme, as shown in Fig. 15. Both cache partitioning schemes use our proposed access-aware policies, and only the writes in STT-RAM region are counted. The write counts of four SRAM banks are not shown. It can be observed from Fig. 15 that the write counts distribution among banks is not very uniform due to the fact that the write counts to each partition are unbalanced. To increase the efficiency of proposed access-aware policies, it is necessary to dynamically change the size of partitions according to the write counts to each partition.

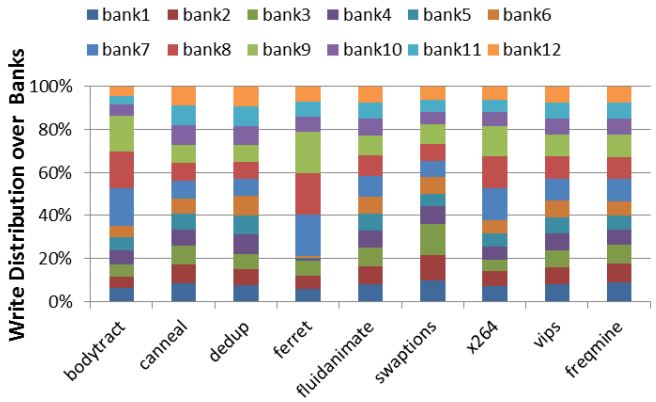

Fig. 15. Unbalanced write count distribution on each bank using the equal-partition scheme.

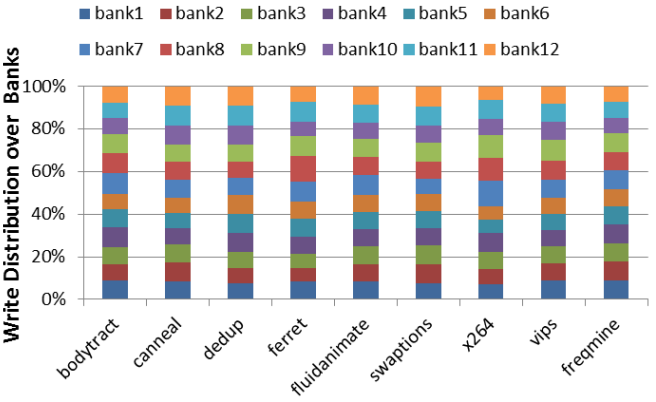

Fig. 16. Improved write count distribution on each bank using the proposed dynamic cache partitioning scheme.

Fig. 16 shows the improved write count distribution on each bank with the proposed dynamic cache partitioning scheme and access-aware policies. Compared with equal-partitions, our dynamic cache partitioning scheme can mitigate the unbalance in write count distribution among the cache banks, especially in the case of *bodytract*, *ferret*, and *x264* benchmarks that exhibit significant unbalanced writes among each core.

E. Sensitivity Analysis of Hybrid Bank

In the hybrid cache bank, the SRAM capacity can affect the write count distribution of the proposed methodology. Increasing the SRAM size can reduce write counts distributed to the STT-RAM region. However, increasing the SRAM size will lead to more energy consumption due to the high SRAM leakage energy. Therefore, we analyze how the results change when using different SRAM sizes in a local hybrid bank.

We first apply a local hybrid bank, access-aware policies, and the equal-partition scheme to do the sensitivity analysis of the hybrid bank. Fig. 17 shows the distribution and standard deviation of write counts with increasing SRAM capacity. The sizes of the SRAM region in the local hybrid bank change from 0 to 256 kB. It can be seen that as the SRAM sizes in the local hybrid bank increase, the write pressure of the

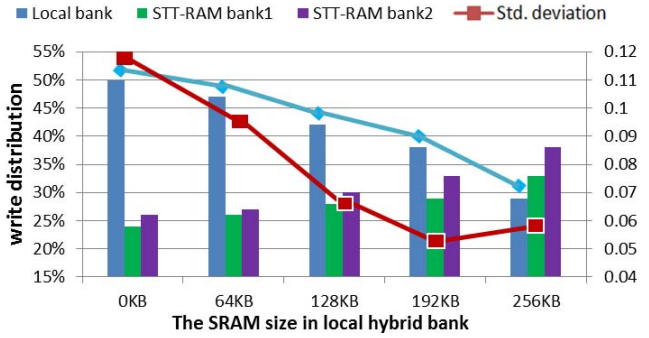

Fig. 17. Analysis of write counts on local bank and standard deviation with access-aware policies and the equal-partition scheme.

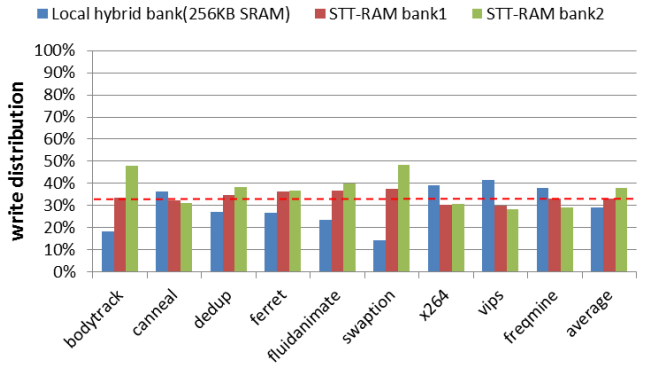

Fig. 18. Unbalanced write count distribution on different banks (the write counts on local bank are overreduced), with access-aware policies and the equal-partition scheme.

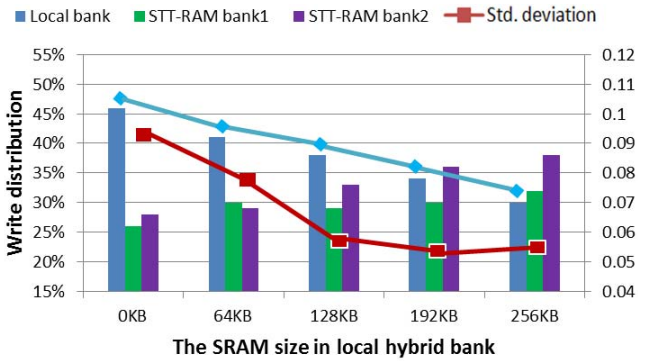

Fig. 19. Reduced write counts on local bank and standard deviation with access-aware policies and a dynamic cache partitioning scheme.

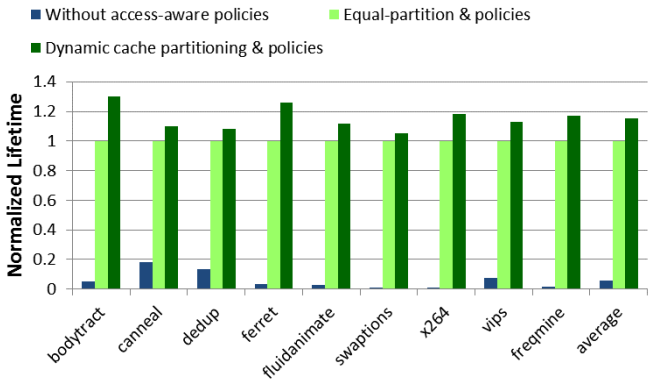

Fig. 20. Comparison of lifetime between the baseline and the proposed wear leveling scheme.

local hybrid bank decreases. However, the standard deviation of the write counts distribution is lowest when the SRAM size is 192 kB, and when the SRAM size increases from 192 to 256 kB, the standard deviation is not reduced. This means that the write distribution is not improved when the SRAM size increases from 192 to 256 kB. The detailed write distribution for each benchmark when the SRAM size in the hybrid bank is 256 kB is shown in Fig. 18. It can be observed that the write pressure on the local hybrid bank is lower than 33% for benchmarks *bodytrack*, *dedup, ferret, fluidanimate*, and *swaption*, which means the write pressure is overreduced. As a result, the write pressure on the nonlocal banks is larger than that of the local banks for these benchmarks, and the unbalanced write distribution is not improved.

Next, we apply the proposed dynamic cache partitioning scheme and compare the distribution and standard deviation of write counts in different hybrid banks, as shown in Fig. 19. It can be observed that compared with Fig. 17, the write count ratio on the local hybrid bank is lower except when the hybrid bank has 256-kB SRAM. The average deviation of write count distribution is also significantly reduced when using the hybrid banks with 0, 64, and 128-kB SRAM. Therefore, when the SRAM size in the hybrid bank is increased, the write pressure on the local bank can be reduced and the unbalanced write distribution is improved. However, when the SRAM size in the hybrid bank is larger than 128 kB, the reduction of the standard deviation is minimal, which means using larger SRAM in hybrid banks cannot achieve significant improvement in regard to balancing the write count distribution. This is because the write distribution is already quite balanced. Therefore, the hybrid banks with 128-kB SRAM are used in our design because they have lower energy consumption and balanced write distribution.

F. Lifetime Comparison

In this paper, we assume the endurance limit of an STT-RAM cell is 4×10^{12}. The STT-RAM cell fails and cannot be used if its write counts exceed 4×10^{12} times during the simulation. To evaluate the lifetime of the STT-RAM region, we repeat the behavior of the write distribution at each benchmark, and the system fails if the total number of defective lines is greater than 16. Fig. 20 compares the estimated lifetime of a baseline hybrid cache without access-aware policies, a hybrid cache with an equal-partition and access-aware policies, and a hybrid cache with the proposed dynamic cache partitioning scheme and access-aware policy. The lifetime values are normalized to the case when the equal-partition and access-aware policies are used by each benchmark. The normalized lifetime of the baseline cache without access-aware policies is lower than 20% of that of the equal-partition cache with access-aware policies. Moreover, the hybrid cache with our dynamic cache partitioning scheme and access-aware policies can increase lifetime by 15% over

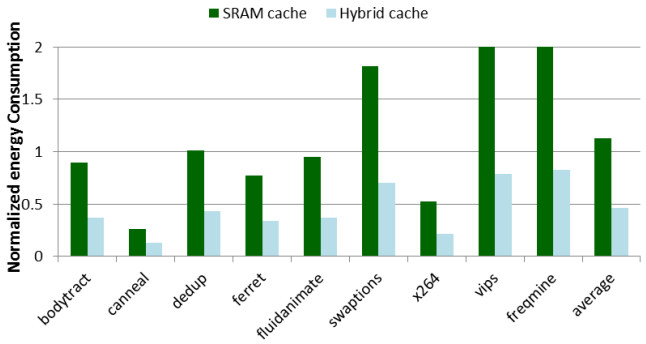

Fig. 21. Energy consumption for SRAM and hybrid cache.

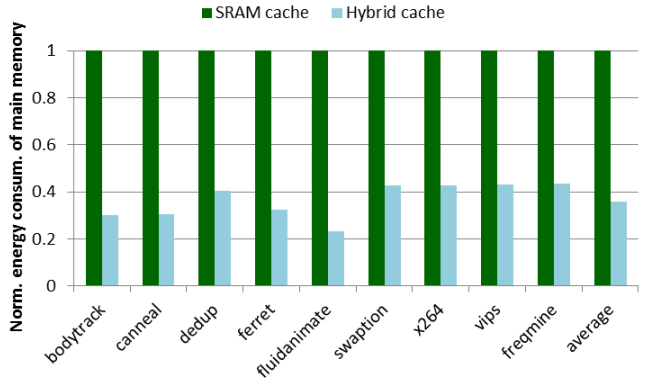

Fig. 22. Energy comparison of main memory for the SRAM and hybrid cache.

the equal partition scheme on average. With our access-aware policies and dynamic cache partitioning scheme, the lifetime of the proposed hybrid cache is 89 times higher than that of the baseline cache without access-aware policies on average.

G. Energy Consumption Comparison

Fig. 21 compares the energy consumption of a SRAM L2 cache and a hybrid L2 cache. The energy consumption is normalized to the maximum energy value in all benchmarks, which happens on benchmark *vips*. The SRAM L2 cache consists of sixteen SRAM banks and provides a total of 8 MB of capacity, which has a similar area to that of the proposed hybrid L2 cache. It can be seen that with our proposed access-aware policies and dynamic cache partitioning scheme, the total energy consumption achieves significant reduction due to the low leakage energy feature of the STT-RAM. The average energy reduction is about 58%. Note that although extra access may occur when migrating less-written data from SRAM to STT-RAM in our SRAM read aware policy, the overall energy is reduced since STT-RAM is used as the main part of hybrid cache.

Fig. 22 compares energy consumption in the main memory with SRAM cache and with the proposed hybrid cache. The energy consumption of the main memory with the proposed hybrid cache is normalized to that of the main memory with the SRAM cache. It can be seen that there is approximately 64% energy reduction on average. This is because with the higher capacity of the hybrid L2 cache, the references to main memory are decreased, and the main memory energy consumption is reduced as well.

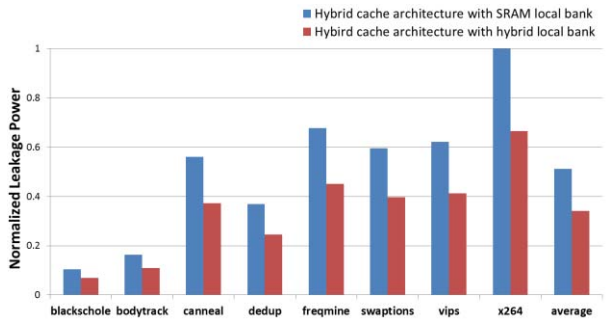

Fig. 23. Normalized leakage comparison between hybrid cache with SRAM local banks and with hybrid local banks.

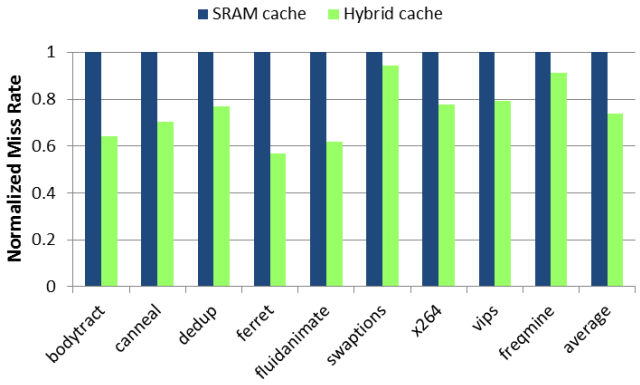

Fig. 24. Access miss rate for SRAM and hybrid cache.

H. Leakage Power Comparision Between Hybrid Cache With SRAM Local Banks and Hybrid Cache With Hybrid Local Banks

On-chip memory power is a major source of power consumption in current system and reducing on-chip memory power consumption is a critical issue. The main advantage of using hybrid local banks is reducing leakage power since more SRAM is replaced with STT-RAM. Fig. 23 compares the leakage power of the traditional hybrid cache architecture that uses SRAM local banks (eight SRAM and eight STT-RAM banks) and the proposed hybrid cache architecture (four SRAM, eight STT-RAM, and four hybrid local banks). It can be seen that the leakage power of the proposed hybrid cache is 33.4% lower than the traditional hybrid cache architecture.

I. Performance Evaluation

Since the capacity of the hybrid cache is about three times larger than that of the SRAM cache, the access miss rate is reduced, as shown in Fig. 24. The average reduction in the miss rate across all the benchmarks is about 26%, which can improve the system performance due to the lower references to main memory. However, the longer write latency of STT-RAM also causes a system performance penalty. Fig. 25 compares the normalized performance between the SRAM L2 cache and the hybrid L2 cache with dynamic cache partitioning and access-aware policies. It can be seen that the hybrid cache has better performance at five benchmarks (the average improvement is about 8.5%) and worse performance at four

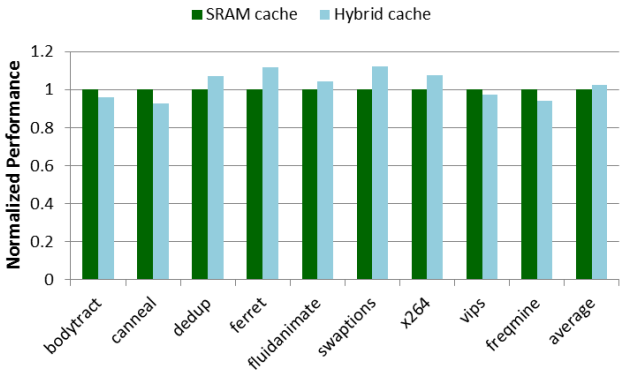

Fig. 25. Normalized performance for SRAM and hybrid cache.

benchmarks (the average slowdown is about 5%). The overall average performance improvement is only 2.5%.

V. Conclusion

In this paper, we proposed a novel hybrid local bank design that consists of SRAM and STT-RAM to reduce the write pressure on local banks. For lifetime extension and performance improvement, two access-aware policies were proposed to increase the write utilization of SRAM and to balance the write distribution over STT-RAM cells. In addition, considering the unbalanced write-pressure over each partition, we proposed a dynamic cache partitioning scheme to further improve the lifetime of the hybrid cache. The experimental results showed that the cache lifetime was improved by 89 times on average compared with the baseline configuration, and it was determined that the hybrid cache can reduce energy consumption by 58% as compared with a SRAM cache.

Acknowledgments

The authors would like to thank Y. Xie of Electrical and Computer Engineering, University of California, and S. Barba and S. Syu of Computer Science and Information Engineering, National Cheng Kung University for their valuable suggestions.

References

[1] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, May 2011.

[2] J. Ahn, S. Yoo, and K. Choi, "DASCA: Dead write prediction assisted STT-RAM cache architecture," in *Proc. IEEE 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2014, pp. 25–36.

[3] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. 17th Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, Oct. 2008, pp. 72–81.

[4] A. Bui *et al.*, "Platform characterization for domain-specific computing," in *Proc. 17th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan./Feb. 2012, pp. 94–99.

[5] J. Cong, K. Gururaj, H. Huang, C. Liu, G. Reinman, and Y. Zou, "An energy-efficient adaptive hybrid cache," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2011, pp. 67–72.

[6] Y. Chen, W.-F. Wong, H. Li, and C.-K. Koh, "Processor caches built using multi-level spin-transfer torque RAM cells," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2011, pp. 73–78.

[7] Y.-T. Chen *et al.*, "Dynamically reconfigurable hybrid cache: An energy-efficient last-level cache design," in *Proc. Design, Autom., Test Eur. Conf. Exhibit. (DATE)*, Mar. 2012, pp. 45–50.

[8] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.

[9] Z. Diao *et al.*, "Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory," *J. Phys., Condens. Matter*, vol. 19, no. 16, p. 165209, 2007.

[10] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler, "A NUCA substrate for flexible CMP cache sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 8, pp. 1028–1040, Aug. 2007.

[11] J.-W. Hsieh and Y.-H. Kuan, "Double circular caching scheme for DRAM/PRAM hybrid cache," in *Proc. IEEE 18th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2012, pp. 469–472.

[12] A. Jog *et al.*, "Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs," in *Proc. 49th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2012, pp. 243–252.

[13] A. Jadidi, M. Arjomand, and H. Sarbazi-Azad, "High-endurance and performance-efficient design of hybrid cache architectures through adaptive line replacement," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2011, pp. 79–84.

[14] L. Jiang, B. Zhao, Y. Zhang, and J. Yang, "Constructing large and fast multi-level cell STT-MRAM based cache for embedded processors," in *Proc. 49th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2012, pp. 907–912.

[15] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *Proc. 10th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Oct. 2002, pp. 211–222.

[16] J. Li, C. J. Xue, and Y. Xu, "STT-RAM based energy-efficiency hybrid cache for CMPs," in *Proc. IEEE/IFIP 19th Int. Conf. VLSI Syst.-Chip (VLSI-SoC)*, Oct. 2011, pp. 31–36.

[17] C. J. Lin *et al.*, "45 nm low power CMOS logic compatible embedded STT MRAM utilizing a reverse-connection 1 T/1 MTJ cell," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Dec. 2009, pp. 1–4.

[18] S. Lee, J. Jung, and C.-M. Kyung, "Hybrid cache architecture replacing SRAM cache with future memory technology," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2012, pp. 2481–2484.

[19] M. Mao, H. Li, A. K. Jones, and Y. Chen, "Coordinating prefetching and STT-RAM based last-level cache management for multicore systems," in *Proc. 23rd ACM Int. Conf. Great Lakes Symp. VLSI (GLSVLSI)*, May 2013, pp. 55–60.

[20] K. J. Nesbit, M. Moreto, F. Cazorla, A. Ramirez, M. Valero, and J. E. Smith, "Multicore resource management," *IEEE Micro*, vol. 28, no. 3, pp. 6–16, May/Jun. 2008.

[21] M. K. Qureshi and Y. N. Patt, "Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches," in *Proc. 39th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2006, pp. 423–432.

[22] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *Proc. IEEE 15th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2009, pp. 239–249.

[23] J. Wang, X. Dong, and Y. Xie, "Point and discard: A hard-error-tolerant architecture for non-volatile last level caches," in *Proc. 49th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2012, pp. 253–258.

[24] Z. Wang, D. A. Jimenez, C. Xu, G. Sun, and Y. Xie, "Adaptive placement and migration policy for an STT-RAM-based hybrid cache," in *Proc. IEEE 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2014, pp. 13–24.

[25] X. Wu, J. Li, L. Zhang, E. Speight, and Y. Xie, "Power and performance of read-write aware hybrid caches with non-volatile memories," in *Proc. Design, Autom., Test Eur. Conf. Exhibit. (DATE)*, Apr. 2009, pp. 737–742.

[26] G. Wu, H. Zhang, Y. Dong, and J. Hu, "CAR: Securing PCM main memory system with cache address remapping," in *Proc. IEEE 18th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2012, pp. 628–635.

[27] S. Yazdanshenas, M. R. Pirbasti, M. Fazeli, and A. Patooghy, "Coding last level STT-RAM cache for high endurance and low power," in *Computer Architecture Letters*, 2013, pp. 1–4.

[28] *CACTI: An Integrated Cache and Memory Access Time, Cycle Time, Area, Leakage, and Dynamic Power Model, Version 5.3*. [Online]. Available: http://www.hpl.hp.com/research/cacti/, accessed May 2013.

[29] *International Technology Roadmap for Semiconductors*, Semiconductor Industries Association, Washington, DC, USA, 2011. [Online]. Available: http://www.itrs.net/

[30] Y.-T. Chen, J. Cong, H. Huang, C. Liu, R. Prabhakar, and G. Reinman, "Static and dynamic co-optimizations for blocks mapping in hybrid caches," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2012, pp. 237–242.

Ing-Chao Lin (M'09–SM'14) received the M.S. degree in computer science from National Taiwan University, Taipei, Taiwan, and the Ph.D. degree from the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, in 2007.

He was with Real Intent, Inc., Sunnyvale, CA, USA, from 2007 to 2009. Since 2009, he has been with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, where he is currently an Associate Professor. His current research interests include very large scale integration design and computer-aided design for nanoscale silicon, low-power reliable system design, and computer architecture.

Jeng-Nian Chiou received the B.S. degree in computer science and engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2012. He is currently pursuing the M.S. degree in computer science and information engineering from National Cheng Kung University, Tainan, Taiwan.

His current research interests include hybrid on-chip memory design and computer architecture.