EECache: A Comprehensive Study on the Architectural Design for Energy-Efficient Last-Level Caches in Chip Multiprocessors

HSIANG-YUN CHENG, MATT POREMBA, NARGES SHAHIDI, IVAN STALEV,
MARY JANE IRWIN, MAHMUT KANDEMIR, and JACK SAMPSON, Pennsylvania State
University
YUAN XIE, University of California at Santa Barbara

Power management for large last-level caches (LLCs) is important in chip multiprocessors (CMPs), as the leakage power of LLCs accounts for a significant fraction of the limited on-chip power budget. Since not all workloads running on CMPs need the entire cache, portions of a large, shared LLC can be disabled to save energy. In this article, we explore different design choices, from circuit-level cache organization to microarchitectural management policies, to propose a low-overhead runtime mechanism for energy reduction in the large, shared LLC. We first introduce a slice-based cache organization that can shut down parts of the shared LLC with minimal circuit overhead. Based on this slice-based organization, part of the shared LLC can be turned off according to the spatial and temporal cache access behavior captured by low-overhead sampling-based hardware. In order to eliminate the performance penalties caused by flushing data before powering off a cache slice, we propose data migration policies to prevent the loss of useful data in the LLC. Results show that our energy-efficient cache design (EECache) provides 14.1% energy savings at only 1.2% performance degradation and consumes negligible hardware overhead compared to prior work.

Categories and Subject Descriptors: B.3.2 [Memory Structures]: Design Styles

General Terms: Cache Memories

Additional Key Words and Phrases: Cache, power management, energy efficiency

ACM Reference Format:

Hsiang-Yun Cheng, Matt Poremba, Narges Shahidi, Ivan Stalev, Mary Jane Irwin, Mahmut Kandemir, Jack Sampson, and Yuan Xie. 2015. EECache: A comprehensive study on the architectural design for energy-efficient last-level caches in chip multiprocessors. *ACM Trans. Architect. Code Optim.* 12, 2, Article 17 (July 2015), 22 pages.

DOI: <http://dx.doi.org/10.1145/2756552>

1. INTRODUCTION

The power consumption of modern chip multiprocessors (CMPs) has become a primary design constraint in scaling performance with each process generation. Even

This article is an extension of the conference paper entitled “EECache: Exploiting Design Choices in Energy-Efficient Last-Level Caches for Chip Multiprocessors” [Cheng et al. 2014] in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, 2014.

This work is supported in part by NSF grants 1500848, 1461698, 1213052, and 1017277 and the Department of Energy under Award Number DE-SC0005026.

Detailed information about this project can be found at <http://www.ece.ucsb.edu/~yuanxie/projects/ASKS/>.

Authors' addresses: H.-Y. Cheng, M. Poremba, N. Shahidi, I. Stalev, M. J. Irwin, M. Kandemir, and J. Sampson, Computer Science and Engineering Department, Pennsylvania State University; emails: hoc5108@cse.psu.edu, mrp5060@psu.edu, nxs314@cse.psu.edu, ids103@psu.edu, {mji, kandemir, sampson}@cse.psu.edu; Y. Xie, Electrical and Computer Engineering Department, University of California at Santa Barbara; email: yuanxie@ece.ucsb.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2015 ACM 1544-3566/2015/07-ART17 \$15.00

DOI: <http://dx.doi.org/10.1145/2756552>

Fig. 1. (a) On-chip power consumption of a 16-core system with 16MB LLC and (b) the energy breakdown of the LLC with high (1.0) and low (0.1) access frequency. The power consumption is calculated by McPAT [Li et al. 2009], and the energy breakdown is derived by CACTI [Muralimanohar et al. 2009] using 32nm high-performance transistors with parallel data lookup and area-optimal circuits.

Fig. 2. (a) Performance under different L3 sizes relative to 16MB L3 and (b) phase-dependent memory footprints. The fraction of required cache varies over time within an application as well as among applications.

though Moore's law continues to provide increasing transistor counts, the limited on-chip power budget restricts the percentage of active transistors [Venkatesh et al. 2010; Esmaeilzadeh et al. 2011; Taylor 2012; Goulding-Hotta et al. 2012; Allred et al. 2012]. In recent years, an increasing percentage of those transistors are invested in the large last-level caches (LLCs) utilized to bridge the gap between fast CPU cores and slow off-chip memory accesses. Specifically, LLCs occupy as much as 50% of the chip area and contribute to a significant amount of the chip's leakage power [Kurd et al. 2010; Naffziger et al. 2006; Wendel et al. 2010; Wilkerson et al. 2010]. As shown in Figure 1, a 16MB LLC consumes about 27% of on-chip power in a 16-core system, with leakage power dominating the LLC's power consumption. Hence, managing the power consumption of LLCs has become an important design issue for future CMPs.

The high leakage power of the LLC comes from its large size, and its size comes from conservative design-time choices that aim to accommodate most applications' memory footprints. However, not all workloads running on CMPs need the entire cache during their execution. Figure 2(a) illustrates the variable sensitivity of workloads to changes in LLC capacity on a 16-core system. On the x-axis are multiprogrammed workloads composed of benchmarks with different demands on capacity (see Section 6 for workloads and simulation details). For example, workloads LL1 and LL2 do not benefit from a larger capacity, while the performance of TH1 and TH2 improves significantly when a larger LLC is employed. Further, the required cache size may also vary with different program phases, as shown in Figure 2(b). When the required cache size is smaller, some parts of the LLC can be disabled to reduce leakage power. In Figure 2(a), for example, if a 5% performance degradation is acceptable, more than half of the LLC can be disabled to save power in all but two workloads.

Fig. 3. Physical design of (a) uniform cache architecture (b) zoomed in portion of SRAM subarray and (c) slice-based cache organization.

In this article, we propose low-overhead runtime mechanisms to manage LLC power consumption in CMPs. We first introduce a slice-based cache organization that requires only minimal circuit overhead to shut down parts of the LLC. Based on this slice-based cache organization, we next propose a low-overhead approach to monitor cache access behavior and explore different choices of when to power off/power on cache slices to maximize power savings with only negligible performance degradation. Specifically, we consider three important and complementary metrics that guide our slice turn-on and turn-off decisions: **utilization**, **hotness**, and **the distribution of dirty cache lines**. Using these metrics, we make the following major contributions:

- We propose *EECache*, an energy-efficient LLC design that explores the design space from circuit-level cache organization to architectural-level management.
- We analyze the cache access patterns that determine the power-off opportunity, including the spatial access behavior captured by active cache footprint, the temporal access behavior monitored by data reuse frequency, and the writeback penalty of dirty cache lines.
- When turning off parts of the LLC to save power, we utilize low-overhead techniques to monitor the cache access behavior and power-off/power-on coarse-grained cache slices. Detailed analysis shows that our approach incurs significantly less overhead than prior work.
- Our evaluation using both multiprogrammed and multithreaded workloads reveals that considering one metric alone may not be effective across workloads. By taking advantage of the strengths of different metrics, we propose a comprehensive approach that provides 14.1% energy savings with only a 1.2% performance degradation on average.

2. SLICE-BASED LOW-POWER CACHE ORGANIZATION

We craft our design and power management mechanisms starting from the physical implementation of a uniform cache architecture [Muralimanohar et al. 2009; Thoziyoor et al. 2007; Balasubramonian et al. 2011; Kurd et al. 2010; Rusu et al. 2009; George et al. 2007]. This allows us to balance power reduction with area overhead. Figure 3 shows the cache organization along with a section of an SRAM subarray in a four-way cache. Data and tag arrays are designed in the same fashion, with a tag array acting similarly to a data array with much smaller line sizes. A uniform cache consists of several smaller subarrays of SRAM cells, shown as squares in Figure 3(a), in order to optimize the total wire RC delay and improve energy. An H-tree design [Muralimanohar et al. 2009; Thoziyoor et al. 2007; Balasubramonian et al. 2011], labeled “data out” on

the figure, provides equal wiring distance to all subarrays in the cache. Typically, a cache access places an entire set of data in one global row; that is, the set spans multiple subarrays in the horizontal direction, highlighted in gray. Furthermore, the bits of each way can be interleaved at the bit level, resulting in less wiring needed on each subarray output, designated by the black cache line striped across all four gray subarrays. The bit-interleaved design can also help to improve reliability [Kim et al. 2007; Rusu et al. 2006] and thermal efficiency [John et al. 2005; Hu et al. 2008]. If the subblock predecoding technique is employed, each cache access can be limited to a particular subarray to further reduce the dynamic power [John et al. 2005; Hu et al. 2008]. At the die layout level, the bitlines providing way data run perpendicular to the wordlines (WL) selecting data as well as the power (Vdd) and ground (gnd) rails, as shown in Figure 3(b).

This results in a large tradeoff between area overhead and the granularity at which the cache can be turned off to save power. Since the power rails are routed perpendicular to bitlines, they span multiple ways in the same set. This makes it difficult to turn off single ways without either adding multiple wire routes for different ways or rerouting wires parallel to the bitlines. Both of these methods would result in higher area overheads ($>10\%$ from our analysis using CACTI [Muralimanohar et al. 2009]), as they increase the total area of all subarrays in the cache. For example, in Figure 3(a), additional wire routing is required to power gate only one fourth of a single subarray when turning off way0, resulting in larger subarrays. The other option is to force a few ways, rather than all ways, to reside in a single subarray. Figure 3(c) shows an example that forces each subarray to store data from only a single way. Forcing a subset of ways into a single subarray increases the width of the H-tree wire routing between the subarrays. However, depending on the number of ways constrained to a single subarray, the size of the H-tree only increases closest to the subarray output. That is, the size of the H-tree near the center of the cache, closest to the cache output, will be the same size. By utilizing CACTI [Muralimanohar et al. 2009] to analyze the area overhead, we found that the wider H-tree only incurs less than 0.5% additional area overhead, which is much smaller than the area overhead of larger subarrays in way-based shutdowns.

In this work, we utilize the second approach of constraining ways into subarrays, an organization we call *slice*. A slice is a generically sized shut-down granularity that may range from one to all ways in a cache. Data for each of the ways in a slice are placed in the same subarray or group of subarrays sharing a sleep transistor. This work chooses a slice size of 1/16 of the total cache ways: four-ways in our experimental design. Such design allows us to turn off entire subarrays. This results in lower overhead due to both (1) subarray sizes remaining static and (2) the need for fewer power gating transistors in the cache, creating the opportunity for higher leakage power reduction.

3. RELATED WORK

Many prior studies proposed different techniques to reduce the dynamic or leakage power consumption of on-chip caches. For dynamic energy reduction, Ghosh et al. [2009] utilized a segmented counting Bloom filter for each cache way to reduce the number of way lookups. Park et al. [2011, 2012] presented a partial tag-enhanced bloom filter to improve the way prediction accuracy and exploited temporal locality to reduce tag comparison. For lower LLC access energy, Ahsan et al. [2011] proposed methods to avoid accessing SRAM columns whose contents are all 1s or all 0s. In this article, we target reducing the leakage power of LLCs, as leakage power dominates the power consumption of LLCs.

In order to reduce the leakage power consumption of on-chip caches, several prior studies have designed caches by utilizing low-leakage memory technologies [Swaminathan et al. 2012; Jadidi et al. 2011; Chen et al. 2012; Tsai et al. 2014] or

proposed circuit-level methods [Flautner et al. 2002; Powell et al. 2000] for conventional SRAM caches. Some of them [Jadidi et al. 2011; Chen et al. 2012] proposed hybrid cache architecture to partially replace SRAM in LLCs by spin-torque transfer magnetic RAM (STT-RAM) for lower leakage power and designed methodologies to tackle the challenges of high write energy and short lifetime in STT-RAM. Tsai et al. [2014] presented a nonvolatile SRAM (nvSRAM) cache architecture with redundant store elimination to reduce the energy consumption of LLCs. Swaminathan et al. [2012] analyzed the energy efficiency of applying various circuit-level and technology-based methods to LLCs. In this article, we focus on the systems that utilize conventional SRAM to build LLCs. Among the circuit-level methods, drowsy cache [Flautner et al. 2002] and gated-Vdd [Powell et al. 2000] are the two most prevalent techniques to reduce the leakage power of conventional SRAM caches. Flautner et al. [2002] proposed a drowsy cache with state-preserving and low-leakage circuits that rely on voltage scaling and put cold cache lines into drowsy mode to save energy. Drowsy circuits, however, need two supply voltages for each cache line, which incurs design overheads. Powell et al. [2000] introduced a gated-Vdd technique that gates the supply voltage of SRAM cells by exploiting the stacking effect of placing a high-Vt transistor between the SRAM cell and GND. In this article, we use the gated-Vdd technique to power off slices.

Based on these circuit-level power reduction techniques, several architectural approaches have been proposed. Some of the prior studies [Foglia and Comparetti 2014; Bardine et al. 2008, 2010, 2014] target reducing the leakage power in the nonuniform cache architecture [Kim et al. 2003], which partitions a large LLC into subbanks for shorter wire delay. Bardine et al. [2008, 2010] leveraged the nonuniform distribution of most frequently accessed data on banks to adapt the number of active ways to the required cache size. They further evaluated different circuit-level power reduction techniques on dynamic nonuniform cache architecture and proposed a hybrid method that combines gated-Vdd and drowsy circuits to reduce the leakage power [Bardine et al. 2014]. Foglia and Comparetti [2014] designed a power reduction technique that does not rely on application-dependent parameters for dynamic nonuniform cache. In this article, we focus on the uniform cache architecture in conventional systems, but our coarse-grained power management mechanisms can also be applied to the nonuniform cache architecture.

Several prior studies have proposed different power reduction techniques for the conventional uniform cache architecture. Wang et al. [2012] designed methods to power-gate L1 and L2 caches in GPUs when there are no ready threads and memory requests. Ghasemi et al. [2011] used different sizes of SRAM cells for different ways and turned off smaller ways when the workloads demand smaller cache capacity. Kim et al. [2010] proposed to turn off the replicated blocks in private L2 for reducing leakage power consumption. Some techniques [Albonesi 1999; Sundararajan et al. 2012] attempt to exploit partitioning caches by ways and disable useless ways across the whole cache or subgroup of sets [Mittal et al. 2013]. Basu et al. [2013] exploited cache coherence to identify stale data and resize the cache by cache ways. Kadjo et al. [2013] facilitated power gating and migrated high-temporal-locality blocks to live partitions. Kaxiras et al. [2001] invalidated and turned off cache lines in L1 caches when they hold data not likely to be reused in the future. Yang et al. [2002] proposed a hybrid selective-sets-and-ways organization for L1 caches.

These architectural techniques either require offline profiling or high hardware overhead to track the dead blocks or utilize additional hardware that consumes nonnegligible power to monitor cache accesses for each cache way. Moreover, most previous schemes assume a power control granularity of a single way. These fine-grained way-based management schemes do not easily generalize to the slice-based organization, which incurs less power-gating circuit overhead.

Fig. 4. Utilization and hotness of ML1 (with small active footprint but data are frequently reused) and TL1 (with large active footprint but data are rarely reused). Utilization = % of referenced cache lines in a time epoch. Hotness = Hits per slice/Total L3 misses.

Fig. 5. Power-off opportunity captured by (a) utilization (utilization < 0.3 slices), (b) hotness (hotness < 0.075 slices) and (c) static optimal (statically select minimum cache size that incurs $< 5\%$ performance degradation) in a 16MB LLC.

4. METRICS OF INTEREST

In order to save energy by disabling cache slices, we need to exploit variability in cache size requirements. There are three main factors that can be used to make slice turn-on/turn-off decisions: utilization, hotness, and the distribution of dirty cache lines.

4.1. Utilization

Ideally, the cache capacity should be large enough to fit the active cache footprint of workloads (i.e., the unique cache lines referenced in a time epoch), and this indicates the utilization of the cache. Figures 4(a) and 4(b) show the utilization of two types of workloads over their executions. We define the utilization as the percentage of cache lines that are referenced in a time epoch in each slice.¹ Different colors indicate different utilization levels. As shown in the figure, ML1, which is composed of applications with low and medium-size memory footprints, has low utilization in L3, while the cache is highly utilized when running TL1 as it is composed of thrashing benchmarks that occupy the entire L3 and low-utilization benchmarks. Furthermore, the utilization varies across different cache slices and time epochs.

Low-utilization slices represent potential power-off opportunities, as disabling these slices would incur few additional cache misses, and thus negligible performance degradation. For example, Figure 5 shows that if we shut down slices with utilization less than 30%, we can disable 50.9% and 95.9% of the LLC on average to save the leakage power for SPEC2006 and PARSEC workloads, respectively. Utilization alone can capture the power-off opportunity of most of the workloads. However, if referenced cache

¹Section 5 details how to efficiently approximate utilization.

Fig. 6. Distribution of dirty blocks at different workloads and epochs.

lines are seldom reused, such as in TL1, TL2, TH1, and TH2, utilization misses some power-off opportunities. This observation motivates us to consider additional metrics.

4.2. Hotness

In addition to the active cache footprint, the access frequency of the stored data is also important when capturing the power-off opportunity. Referenced lines in the cache may be used only once or be reused multiple times during their lifetime. Disabling a cache slice with many frequently accessed cache lines would incur a higher number of cache misses (even if the slice utilization is low) than shutting down a seldom reused cache slice. In this article, we define the hotness of a cache slice as the number of hits to a cache slice divided by the total number of LLC misses in a time epoch. Thus, the hotness implies the increase in the cache miss rate if the slice is disabled.

Disabling cold slices provides different power-off opportunities than disabling low-utilization slices. Figures 4(c) and 4(d) show the hotness of two types of workloads. The active footprint of ML1 is small, as shown in Figure 4(a), but the referenced cache lines are highly reused, as shown in Figure 4(c). Disabling only cold slices for ML1 would lose considerable power-off opportunities provided by the small active footprint. On the other hand, TL1 references a large number of cache lines in each epoch, as indicated by Figure 4(b), but these referenced cache lines are seldom reused, as illustrated in Figure 4(d). Therefore, disabling cold slices for TL1 may provide higher power savings than disabling low-utilization slices. Figure 5 illustrates that if we shut down the cold slices, such as the slices with hotness less than 7.5%, we can disable 36.7% and 68.6% of the LLC on average for SPEC2006 and PARSEC, respectively. For workloads with a large but seldom reused cache footprint, such as TL1, TL2, TH1, and TH2, the hotness of slices can better capture the power-off opportunity than utilization alone.

4.3. Writeback of Dirty Data

When a cache slice in the LLC is turned off, the dirty cache lines in the slice need to be written back to active memory. Figure 6 shows that the fraction of dirty cache lines varies across different workloads and cache slices. Disabling a cache slice with a higher number of dirty cache lines would reduce power savings, since the slice can only be powered down once all the dirty cache lines have been written back. Furthermore, the large number of writes would frequently fill up the write buffer in the memory controller and could delay critical reads. Thus, a cache slice with a lower number of dirty cache lines should be chosen among the slices with the same level of utilization or hotness when deciding which slice should be disabled to save energy.

In summary, cache utilization indicates spatial access behavior, while the hotness of cache slices indicates temporal access behavior. Low-utilization cache slices can be disabled to save the leakage power of the unused cache capacity, while cold slices can be turned off when the referenced data are seldom reused to further increase the power-off opportunity. Also, when choosing which slices to power off, the number of dirty lines should play an important role. The previous discussion suggests that an

Fig. 7. Overview of EECache and power management process.

ideal slice power-off/on strategy should consider all these metrics at the same time. In Section 5, we will describe how to capture these three characteristics with small hardware overhead and exploit these characteristics to design the power-off, power-on, and data migration strategies for improving the energy efficiency of the LLC.

5. EECACHE

In this article, we propose low-overhead methods to monitor the cache access behavior and design the power-off, power-on, and data migration policies accordingly. Figure 7 shows the overview of the proposed scheme. In the cache controller, we add a power management unit (PMU) to determine the power state of each cache slice at each time epoch. The PMU monitors the cache behavior, including the utilization, hotness, and dirty status in each slice, and uses this information to guide the power-off, power-on, and data migration policy.

At the beginning of each epoch, that is, time t in the figure, the PMU collects the cache access status from the previous epoch. Based on the cache access behavior, the PMU first decides whether the workload will benefit from a higher capacity. If there is no need to turn on powered-off slices, the PMU then checks whether some of the slices can be turned off to save even more power. Before turning off the victim slices selected by the power-off policy, the PMU decides whether each dirty block in the victim slices should be written back to the main memory or migrated to other slices. The clean blocks also need to be either discarded or migrated to other power-on slices. During the transition phase, the valid cache lines that have not been flushed can still be accessed, while the new cache lines are not allowed to be inserted into the victim slices and can only be placed into the active portion of the LLC. After all the blocks in the victim slices are flushed, that is, at time $t + T_{\text{migrate}}$ in Figure 7, the victim slices can be turned off to save leakage power until the end of the epoch. Next, we explain how to monitor the cache access behavior with low-overhead hardware and describe our power management policies in detail.

5.1. Monitoring Cache Access Behavior

As discussed in Section 4, the benefits and drawbacks from powering off slices are closely related to the utilization, hotness, and distribution of dirty cache lines. Figure 8 shows how to dynamically monitor the cache access behavior with small hardware overhead. To capture hotness, we use two 32-bit counters (CHc and DHc) to count the number of hits to each clean and dirty cache line in the slice. The number of LLC misses is captured by a global counter (Mc) for all the cache slices. To monitor the distribution of dirty blocks, we utilize a counter (Dc) to count the number of dirty cache lines in each slice. For profiling the utilization, we develop a sampling-based method that is inspired by a prior set-sampling approach [Qureshi et al. 2006] to reduce the hardware overhead. The key idea behind set sampling is that the behavior of the cache can be

Fig. 8. Profiling cache access behavior in each cache slice. Uc: utilization counter; CHc/DHc: clean/dirty hit counter; Dc: dirty counter; Rs: sampling rate of utilization.

Fig. 9. Correlation coefficient between the estimated utilization using set sampling and the actual utilization for a 1MB L3 cache slice with 4,096 sets.

approximated by sampling only a few sets. Instead of using 1 bit per cache line, we sample the utilization of only $1/R_s$ sets to estimate the overall utilization in each cache slice. When a cache line is inserted into the sample set, the corresponding utilization bit is set to one. The utilization counter (Uc) of the slice is then increased by one. When the cache line is evicted or invalidated, the mapped utilization bit is reset to zero and Uc is decreased by one. All the utilization bits are reset to zero at the beginning of each epoch to filter the stale data. Suppose that there are N_w ways and N_s sets in each slice; we can then calculate the utilization and hotness by the following equations at the end of each epoch:

$$\text{SliceUtilization} = U_c / (N_w * N_s * (1/R_s)) \quad (1)$$

$$\text{SliceHotness} = (CHc + DHc) / Mc. \quad (2)$$

Figure 9 illustrates that sampling only $1/64$ sets in each cache slice is enough to achieve a higher than 80% correlation coefficient in a 1MB LLC cache slice with 4,096 sets, when compared to the real utilization (1 utilization bit per cache line). Thus, we chose to use the $1/64$ sampling rate for the experiments in Section 7.

5.2. Power-Off Policy

Based on the monitored cache access behavior, we develop and analyze three different power-off policies:

5.2.1. Utilization-Based Policy (Uoff). When the active cache footprint is small and a slice is underutilized, disabling the slice can save leakage power while incurring only negligible performance degradation. Motivated by this, the Uoff policy selects the power-off

Fig. 10. Flow chart of the UHDOff power-off policy.

victims from the slices with the utilization (calculated by Equation (1)) less than a threshold, U_{th} . Among the low-utilization power-off victims, the slices with lower utilization are chosen to be turned off. At most N_{off} slices are powered off at an epoch to avoid burst of writebacks. Note that a higher U_{th} and a higher N_{off} would aggressively provide higher power savings, but the performance degradation would also increase. We analyze the impact of different U_{th} and N_{off} settings and find that 30% and 4 are good values for U_{th} and N_{off} , respectively.

5.2.2. Hotness-Based Policy (H_{off}). Disabling a cold slice with a lower number of frequently referenced cache lines to save leakage power would incur a lower number of cache misses than powering off a hot slice. Thus, the H_{off} policy selects the power-off victims from the slices with the hotness (calculated by Equation (2)) less than a predefined threshold, H_{th} . With less than H_{th} hotness, the increase in cache miss rate would possibly be less than H_{th} when the slice is powered off. Among the cold power-off victims, the slices with lower hotness are chosen to be turned off, with the restriction that at most N_{off} slices are powered off at an epoch to avoid write bursts. A higher H_{th} setting would turn off more cache slices to save power but also increase the performance penalty. We analyze the impact of different H_{th} settings and find that an H_{th} equal to 7.5% represents a good tradeoff for the system we evaluated.

5.2.3. Utilization+Hotness+Dirty-Aware Policy (UHDOff). As observed in Section 4, using the utilization alone to determine the power state would lose some power-off opportunities when the referenced cache lines are seldom reused. The hotness characteristic can help to identify these seldom reused slices, and powering off these slices can further reduce the leakage power. Moreover, with similar utilization and hotness, the slice with fewer dirty cache lines would incur a lower writeback penalty before shutting down the cache slice. Therefore, the UHDOff policy considers all three factors when deciding when and which slice should be disabled to save power, as illustrated in Figure 10 and Algorithm 1. We first select the power-off victims from the slices with less than U_{th} utilization (lines 6 to 13). If there is no low-utilization slice, we instead select the power-off victims from the slices with less than H_{th} hotness (lines 14 to 23). Among the power-off victims, at most N_{off} slices with fewer dirty cache lines are chosen to be turned off (lines 25 to 32). In order to maintain the inclusion property, at least $MinOn$ slices stay at the power-on state to guarantee that the LLC size is larger than the upper-level caches. In this algorithm, we first consider turning off the low-utilization slices, then the cold slices. The reason is that the utilization fails to capture the power-off opportunity of the seldom reused slices for the workloads with large memory footprint, while the hotness metric can help to identify these less frequently accessed slices, as illustrated in Section 4. The dirty metric is then used to select the slice that incurs lower writeback overhead. We analyze the impact of different threshold settings and empirically set $U_{th}=30\%$, $H_{th}=7.5\%$, and $N_{off}=4$.

5.3. Power-On Policy

After some slices are powered off, the power consumption is decreased but the cache misses may increase due to the smaller LLC size. Since the cache access behavior changes in different program phases, we need to determine whether the workload

ALGORITHM 1: Power-off policy

```

1:  $N$ : number of slices in LLC
2:  $S_{on}$ : number of powered-on slices
3:  $victim[N]$ : power-off victim slices
4:  $Pstate[N]$ : power state of each slice; 0 = on; 1 = off
5:
6:  $V_c = 0$ ; //Initialize number of victim slices
7: for  $i = 0$ ;  $i < N$ ;  $i++$  do
8:    $u_i = U_{c_i} / [Nw * Ns * (1/Rs)]$ ; //utilization of slice  $i$ 
9:   if  $u_i < U_{th} \wedge Pstate[i] = 0$  then
10:      $victim[V_c] = i$ ;
11:      $V_c++$ ;
12:   end if
13: end for
14: if  $V_c = 0$  then
15:   //if there's no low-utilization slice, find cold slices
16:   for  $i = 0$ ;  $i < N$ ;  $i++$  do
17:      $h_i = (CH_{c_i} + DH_{c_i}) / Mc$ ; //hotness of slice  $i$ 
18:     if  $h_i < H_{th} \wedge Pstate[i] = 0$  then
19:        $victim[V_c] = i$ ;
20:        $V_c++$ ;
21:     end if
22:   end for
23: end if
24: //shutdown at most  $N_{off}$  slices to avoid significant increase in cache misses
25: for  $i = 0$ ;  $i < \min(V_c, N_{off})$ ;  $i++$  do
26:   //guarantee at least  $MinOn$  slices are power-on
27:   if  $S_{on} > MinOn$  then
28:     //shutdown the slice with  $i$ th lowest number of dirty cache lines
29:      $Pstate[GetMinDc(victim, i)] = 1$ ;
30:      $S_{on} = S_{on} - 1$ ;
31:   end if
32: end for

```

would benefit from a larger LLC at the beginning of each epoch. In this article, we develop and discuss three different power-on policies:

5.3.1. Utilization-Based Policy (U_{on}). When the utilization of the LLC is extremely high, it is possible that the workload needs a larger LLC capacity to better fit the active memory footprint. Therefore, the U_{on} policy turns on N_{off} slices to improve performance when the utilization of the entire active portion of the LLC is higher than a predefined threshold U_{onTh} . The utilization of the entire active portions of the LLC is calculated by

$$ActiveUtilization = \frac{\sum_{i=active-slice} U_{c_i}}{S_{set} * S_{way} * S_{on} * (1/Rs)}. \quad (3)$$

The S_{on} in the equation represents the number of powered-on slices in the LLC. We turn on N_{off} slices, that is, the maximum number of slices that are powered off in the previous epoch, instead of the entire cache to avoid sacrificing too much power savings for performance improvement. With a higher U_{onTh} threshold setting, the disabled slices would remain in the power-off state for a longer time, but the performance degradation would also increase. We empirically analyze the pros and cons of different U_{onTh} settings and find that a U_{onTh} of 90% performs quite well.

5.3.2. Victim-Hit-Based Policy (VHon). If the number of accesses to the powered-off portion is high, the workload may demand more cache slices to service the cache access requests. In order to detect the demand for a larger cache capacity, we keep the whole tag array powered on to monitor the potential hits to the powered-off portion of the LLC. The tag array of the powered-off slices are called victim tags [Chen et al. 2012; Cong et al. 2011]. These victim tags reuse the same tag array as the original LLC and thus do not create extra storage overhead. The tag of the evicted cache line from the active portion of the LLC would be inserted into the victim tags. A potential hit counter, $Vhit$, is increased by one when a hit occurs to the victim tags. When the hit rate to the victim tags is higher than a threshold, $HonTh$, $Noff$ slices (i.e., the maximum number of slices that are powered off in the previous epoch) are powered on to improve performance. The hit rate of the victim tags is calculated by the number of victim hits divided by the total number of LLC misses, that is, $Vhit/Mc$. Increasing the value of $HonTh$ provides more power savings by turning on the powered-off slices less often, but the performance degradation would also increase. We analyze the impact of different $HonTh$ settings and find that $HonTh = 10\%$ is a good value, respectively.

5.3.3. Dynamic Power-On Granularity (DynOn). Instead of turning on a fixed number of $Noff$ slices at each power-on process, we propose a DynOn policy to dynamically power on a different number of slices according to the potential benefit brought from a larger LLC capacity. The DynOn is based on the VHon policy and utilizes the hit rate of the victim tags to decide how many slices should be powered on for performance improvement. The number of slices to be turned on is determined by the following equation:

$$OnSlice = VHitRate/DonTh = (Vhit * DonTh)/Mc. \quad (4)$$

The $DonTh$ in the equation implies the expected reduction in LLC misses when one slice is turned on. Thus, the number of slices that should be powered on is proportional to the potential reduction in cache misses brought from enabling one single slice. A higher $DonTh$ setting would increase the power savings by powering on a lower number of slices, but the performance degradation would also increase. The $DonTh$ is also sensitive to the power-off/power-on granularity and should be set to a lower value when the LLC is managed in finer granularity, that is, composed of a higher number of slices. In this article, we empirically set the $DonTh$ value to 2% to trade off the power savings and performance degradation.

5.4. Data Migration Policy

Before a slice is disabled to save the leakage power, the data stored in the slice need to be either written back, migrated to other active slices, or discarded, depending on their clean/dirty status and access frequency. The goal of the data migration process is to guarantee data coherency while reducing the miss penalty due to the loss of data in the powered-off slices. In this article, we discuss two different data migration policies:

5.4.1. Cache-to-Memory Policy (CtoM). A conventional method to deal with the data stored in the victim slices is to discard all the clean cache lines and write back all the dirty blocks. Dirty data that is written back may be reused later, incurring additional cache misses. If the dirty cache line is written again after being reinserted into the active portion of the LLC, the flushing of the dirty cache line causes an additional and unnecessary dirty writeback, compared to when the victim slice is not powered off. Thus, the conventional CtoM policy would significantly increase the off-chip traffic and dynamic power consumption.

5.4.2. Hot-to-Cache Policy (HtoC). To reduce the cache miss penalty caused by losing the data stored in the powered-off slices, a possible solution is to migrate useful data

to other active slices. During each migration, a replacement victim is selected from the active portion of the LLC according to the underlying replacement policy, that is, the LRU block in active slices. The valid cache line in the slice to be powered off is then migrated to the active slice after the replacement victim is evicted. Note that the migration of a cache line may incur an additional conflict miss if the evicted replacement victim is reused later. Thus, we choose a HtoC policy to migrate only the cache lines stored in hot slices to other active slices during the transition phase. The hot-clean slices with a high hit rate to clean blocks and hot-dirty slices with a high hit rate to dirty blocks are identified by the following two equations:

$$HotClean : CHc/Mc > Mth \quad (5)$$

$$HotDirty : DHc/Mc > Mth. \quad (6)$$

The Mth is the migration threshold. With lower Mth settings, the LLC would lose less data in the powered-off slices but may incur a higher number of conflict misses and evict more useful data in other active slices. The setting of Mth is also sensitive to the number of slices. When the LLC is composed of a lower number of slices, that is, coarser power-off/power-on granularity, the Mth should be increased to migrate only useful data. We analyze the impact of different Mth settings and empirically set Mth to 4. If a slice is identified as a hot-clean slice, all the clean cache lines are migrated to other active slices when this slice is disabled. Otherwise, all the clean cache lines are discarded. Similarly, the dirty cache lines are only migrated if this slice is identified as a hot-dirty slice. The migration of data between slices is performed in the background and does not delay the demand requests to the cache. When there is a read or write request to the LLC, the data migration would be interrupted. The request would be serviced from the slice to be powered off if the data hasn't been migrated. Otherwise, the request would be serviced from the new data location at other active slices. The data migration process will continue after finishing the service of the demand request. Note that our migration policy is independent of the underlying replacement policy and can be applied to different replacement schemes.

5.5. Summary

In this section, we propose power management strategies for slice-based cache organization. We utilize low-overhead hardware to monitor the cache access behaviors, that is, utilization, hotness, and the amount of dirty cache lines, that are closely related to the power-off opportunity. According to the monitored cache access behavior, we develop and discuss the pros and cons of different power management strategies. The power management policies we explore and the corresponding thresholds are summarized in Table I and Table II, respectively. The thresholds are mainly sensitive to the acceptable performance degradation, while the hotness-related thresholds, such as Hth , $HonTh$, and Mth , are also sensitive to the power-off/power-on granularity. We analyze the impact of different threshold settings and choose the setting listed in Table II to provide better energy efficiency. Our power management mechanism requires only 0.005% storage overhead in a 16MB LLC, as shown in Table III, and is independent of the underlying replacement policy.

6. EXPERIMENTAL SETUP

We evaluate our designs using the cycle-accurate gem5 simulator [Binkert et al. 2011], augmented with McPAT [Li et al. 2009] to evaluate the system power consumption. The McPAT tool is modified to estimate the power of L3 caches. When calculating the power consumption, the leakage power savings, the increase in dynamic power due to additional cache misses, and the additional power consumption of our monitoring

Table I. Summary of Our Power Management Schemes

Power-off policy	Uoff	Utilization-based power-off policy that power off at most $Noff$ lowest-utilization slices with the utilization $< Uth$
	Hoff	Hotness-based power-off policy that power off at most $Noff$ coldest slices with the hotness $< Hth$
	UHDoff	Utilization+Hotness+Dirty-aware power-off policy that selects the power-off victim slices from (1) utilization $< Uth$ slices or (2) hotness $< Hth$ slices if there's no low-utilization slices. Among the power-off victims, at most $Noff$ slices with the lowest number of dirty cache lines are chosen to be powered off.
Power-on policy	Uon	Power-on $Noff$ slices when the overall utilization of the active slices is $> UonTh$
	VHon	Power-on $Noff$ slices when the hit rate to the victim tags of the powered-off slices is $> HonTh$
	DynOn	Dynamically power-on k slices if the hit rate to the victim tags is higher than $k * DonTh$
Data migration policy	CtoM	Discard all the clean blocks and write back all the dirty blocks
	HtoC	Migrate only the hot-clean and -dirty blocks that are likely to be reused to other active slices

Table II. Threshold Settings (the Value in the Table Indicates the Setting That Provides the Best Average Energy Efficiency for Our System)

Threshold	Description	Setting
Rs	Set-sampling rate for utilization	64
Uth	Utilization threshold for power-off	30%
Hth	Hotness threshold for power-off	7.5%
Noff	Maximum number of slices to be powered off at one epoch	4
UonTh	Utilization threshold for power-on	90%
HonTh	Hit rate threshold of victim tags for power-on	10%
DonTh	Hit rate threshold of victim tags for dynamic power-on granularity	2%
Mth	Hotness threshold for data migration	4%

Table III. Storage Overheads

Hardware	Description	Overhead
Uc	Utilization counter per slice	32 bits * 16(slice) = 64B
CHc	Clean hit counter per slice	32 bits * 16 = 64B
DHc	Dirty hit counter per slice	32 bits * 16 = 64B
Mc	Miss counter for entire LLC	32 bits = 4B
Dc	Dirty counter per slice	32 bits * 16 = 64B
Vhit	Hit counter for victim tags	32 bits = 4B
Uarray	Utilization set-sampling array	64(way) * 64(sampled set) = 512B
All		776B/16MB $\approx$ 0.005%

mechanism are all included. The baseline configuration for the CMP system is shown in Table IV.² To evaluate the energy efficiency of our EECache, we also compare it to a state-of-the-art way-based power-gating prior work (PGM) [Kadjo et al. 2013] that utilizes a sampler tag array with program counter information to decide when to power off/power on and migrate useful data in the powered-off ways. The area overhead comparison is carried out using CACTI [Muralimanohar et al. 2009]. When

²EECache can also be applied to a system with private L2 cache. The 10M-cycles Epoch is chosen to be short enough for phase detection but long enough to accommodate write-back and migration overhead.

Table IV. System Configuration

Processor	16 cores; 2GHz; out-of-order; issue width = 4; 192 ROB; 32 LSQ
L1 I&D-cache	Private; 32KB per core; 4-way; 64B line size; 2 cycles
L2 cache	Shared; 4MB total; 16-way; 64B line size; 20 cycles
L3 cache	Shared; 16MB total (1MB per slice); 64-way (4-way per slice) 30 cycles; MSHR = 32; 5 slices are always on for inclusion property
Memory	120 cycles access latency
Epoch	10M cycles

Table V. Workload Characteristic and Classification of the SPEC2006 and PARSEC Programs
(Util = Average Utilization Per Epoch)

Multiprogrammed			Multithreaded	
	Benchmark	L3Util	Benchmark	L3Util
LL1	GemsFDTD,tonto,tonto,povray	3.6%	bodytrack	0.1%
LL2	GemsFDTD,povray,povray,tonto	2.0%	swaptions	0.1%
ML1	dealII,xalancbmk,GemsFDTD,povray	6.8%	blackscholes	0.1%
ML2	gromacs,h264ref,povray,tonto	14.4%	fraqmine	0.7%
HL1	gobmk,omnetpp,GemsFDTD,tonto	12.5%	x264	4.7%
HL2	hmmmer,zeusmp,povray,tonto	52.9%	ferret	7.5%
HM1	gobmk,hmmmer,dealII,gromacs	37.9%	rtview	8.8%
HM2	omnetpp,zeusmp,h264ref,namd	58.3%	dedup	15.4%
TL1	astar,mcf,GemsFDTD,povray	61.9%	fluidanimate	25.9%
TL2	leslie3d,libquantum,povray,tonto	98.6%	canneal	72.0%
TH1	astar,leslie3d,gobmk,hmmmer	97.0%		
TH2	libquantum,sjeng,omnetpp,zeusmp	91.6%		
Type	Description			
L	Low Util in L2 (<60%) and L3 (<30%): GemsFDTD, povray, tonto			
M	High Util in L2 but medium Util in L3 (30–60%): dealII, gromace, h264ref, namd, xalancbmk			
H	High Util in L2 and L3 (>60%): gobmk, hmmmer, omnetpp, zausmp			
T	Thrashing benchmark, Util does not decrease as the core counts decreases: astar, leslie3d, libquantum, mcf, sjeng			

reporting the system energy consumption, the main memory access energy is included and measured by DRAMPower2 [Chandrasekar et al. 2014].

We use a set of SPEC CPU2006 benchmarks for multiprogrammed workloads and PARSEC benchmarks for multithreaded workloads.³ For the SPEC benchmarks, we fast-forward 500M instructions and run in detailed mode for the next 1 billion instructions. All the SPEC benchmarks use the reference input sets. We classify the SPEC benchmarks into four categories, as shown in Table V, according to their cache utilization. We create the multiprogrammed workloads by combining two different categories of applications. The workload combination covers a broad range of cache access behaviors. For the PARSEC benchmarks, we run 1 billion instructions starting from the region of interest (ROI) [Gebhart et al. 2009], using the simlarge input set. When reporting performance results, we use overall throughput ($\sum IPC_i$).

7. EXPERIMENTAL RESULTS

In this section, we first analyze the power-saving and performance impact when different power management policies are applied. We then show that our EECache can provide significant energy savings with small hardware overhead. When comparing

³We choose the set of benchmarks that is portable to our simulation environment.

Fig. 11. (a) System performance and (b) L3 power savings of different power-off policies.

different power management policies, we in order evaluate the energy efficiency of different power-off, power-on, and data migration policies. During the evaluation of the power-off policies, the VHon policy with fixed power-on granularity is applied and all the clean/dirty data in the powered-off slices are discarded/written back to eliminate the impact on the result. When evaluating the power-on policies, the best-performing power-off policy (i.e., UHDOff) is used and there's no data migration when a slice is powered off. The UHDOff policy is also utilized when comparing the data migration policies, and VHon, the power-on policy that provides the highest power savings, is applied. After analyzing the energy efficiency of different management strategies, we evaluate the combination of UHDOff+VHon+HtoC, which is the most energy efficient among the possible strategies, against a way-based power-gating scheme [Kadjo et al. 2013].

7.1. Power-Off Policies

Figure 11 shows the performance degradation and power savings when different power-off policies are applied. As discussed in Section 4, the utilization of caches indicates the spatial access behavior of the workload. Therefore, powering off low-utilization slices (<30% utilization in the figure) can provide about 51% power savings on average, with a less than 2% performance degradation. However, when the access footprint of the workloads is large, such as in TL2, TH1, TH2, and canneal, the utilization-based power-off policy (Uoff) loses some power-off opportunity when the data stored in the LLC are seldom reused. Utilizing hotness as the power-off metric (Hoff) can disable the rarely reused slices in these high-utilization workloads. However, the Hoff policy fails to capture the power-off opportunity when the workload frequently accesses a small amount of hot data, such as in ML1, ML2, HL2, HM1, and ferret. Within similar performance degradations, the UHDOff policy can better capture the power-off opportunity in both types of workloads by disabling the underutilized slices in low-utilization workloads and power-gating the seldom reused slices when the memory footprint occupies the entire cache. On average, the UHD policy can provide 57% and 17% power savings in the LLC and the entire processor, respectively, while incurring only a less than 2% performance degradation. When calculating the power consumption, the leakage power savings, the increase in dynamic power due to additional cache misses, and the additional power consumption of our monitoring mechanism are all included.

Fig. 12. (a) System performance and (b) L3 power savings of different power-on policies.

7.2. Power-On Policies

After some slices are powered off, we need an effective power-on policy to turn on some slices when the cache demands more capacity. We compare three different policies that power on some portions of the cache according to the utilization (Uon), or the potential hits to the powered-off slices (VHon and DynOn), to avoid sacrificing too much performance when saving the leakage power. As shown in Figure 12, the Uon policy powers on too frequently at high-utilization workloads, such as TL2, TH1, and TH2, and eliminates the benefit of powering off the seldom reused slices, provided by the UHDOff policy. The VHon policy provides the highest power savings among the three policies, as it powers on less often at workloads with high utilization but a low data reuse rate, such as TL1, TL2, TH1, and TH2. However, the VHon policy fails to detect the increasing demand on the cache capacity in some workloads, such as ferret and dedup, thus incurring a higher than 5% performance degradation in these workloads. These demands of higher cache capacity can be detected by the DynOn policy that dynamically powers on a different number of slices according to the potential hits to the powered-off portions. Therefore, the DynOn policy is less aggressive at power saving but can provide a less than 5% performance degradation for all the workloads. On average, the combination of UHDOff+VHon and UHDOff+DynOn can provide 57% and 52% L3 power savings, resulting in 21% and 16% savings in system energy, respectively.

7.3. Data Migration Policies

Before disabling some cache slices to save power, the valid data stored in the slices need to be either discarded/written back or migrated to other active slices. Figure 13(a) shows that the HtoC policy, which migrates cache lines to other active slices before shutting down a hot slice, can provide better performance, especially at ferret and dedup, as they experience frequent phase changes and the stored data are highly reused. The performance improvement comes from reducing the additional cache misses caused by losing the data stored in the powered-off slices and the decrease of dirty writebacks, as illustrated in Figure 13(b). By simply using the clean and dirty hit counters in each slice, we can provide an effective prediction of the useful data, even when running a mix of applications in multiprogrammed workloads. The overhead is much smaller than the prior work [Kadjo et al. 2013] that utilizes a large sampling array to store the recently accessed instruction for predicting the usefulness of the data to be flushed.

Fig. 13. (a) System performance and (b) reduction in L3 misses and writebacks to main memory. Performance is normalized to baseline 16MB L3, while L3 misses and writebacks are compared to CtoM.

Fig. 14. System power, energy, and EDP saving of EECache, compared to baseline 16MB LLC.

7.4. Energy Efficiency of EECache

After analyzing the pros and cons of different power management policies, we evaluate the energy efficiency of our EECache. Among the combinations of different power-off, power-on, and data migration policies, we choose to evaluate the most energy-efficient scheme (UHDoff+VHon+HtoC), which considers both utilization and hotness when selecting the slice to be powered off, and power on a fixed granularity (four slices) of cache based on the potential hits to the powered-off slices. The data stored in the hot slices are migrated before being powered off. Figure 14 shows that EECache (UHDoff+VHon+HtoC) provides 14.8% system power savings on average. The power consumption of the whole processor, the main memory, and the additional hardware, such as the utilization sampler and counters, are all included. Since EECache reduces the power consumption with less than 2% performance degradation, the energy consumption of the whole system is also reduced by 14.1%. Although the energy saving is lower at workloads with higher performance degradation, such as TH1 and TH2, EECache still provides 13.4% energy-delay-product (EDP) savings on average.

7.5. Comparing to a Prior Way-Based Approach

We compare EECache (UHDoff+VHon+HtoC) against a state-of-the-art way-based PGM [Kadjo et al. 2013]. PGM uses a high-overhead sampling array that stores the tags of the sample sets and the instructions that recently accessed the blocks. The sample tags are managed in LRU replacement scheme, which brings large overhead in the highly associative LLC. They determine the required cache size by analyzing the hits to the LRU sample tags and blindly choose the ways to be powered off. Thus, it is possible that highly utilized or frequently accessed blocks are selected to be powered off. Therefore, they rely on a high-overhead prediction scheme that hashes the instructions into a counter array to determine whether a cache line is useful and should be migrated

Table VI. Hardware Overheads of PGM and EECache in a 16MB LLC

	PGM	EECache
Storage overhead	103.25KB area = 0.907mm ² (3.3%) leakage = 0.516W (2.5%)	776B area = 0.100mm ² (0.4%) leakage = 0.057W (0.3%)
Gated-Vdd circuit	3.0%	1.7%
Routing overhead	larger subarrays (14.3%)	wider H-tree (0.4%)
Total area overhead	20.6%	2.5%

Fig. 15. (a) System performance and (b) system power savings of prior way-based power-gating (PGM) [Kadjo et al. 2013] and our EECache.

to other active ways during the transition phase. Our EECache design consumes negligible storage overhead (0.4% extra L3 area), which is smaller than the PGM scheme, as shown in Table VI, resulting in less leakage power consumption. Since we use slice-based shutdowns, the gated-Vdd circuit and routing overhead are smaller than the fine-grained way-based shutdowns in PGM. As a result, the overall area overhead of EECache is only 2.5%, much less than the 20.6% overhead in the PGM scheme.

Figure 15 illustrates the system performance and power savings of the prior PGM approach and our EECache. The prior PGM approach relies on the hits to LRU ways to determine the required cache size. However, this scheme does not consider the active footprint of the LLC, thus sacrificing the performance of some high-utilization workloads, such as TH1 and TH2, to provide higher power savings. The PGM also incurs a higher than 5% performance degradation for ferret and dedup, due to the failure to detect the power-on demand in these workloads. Our EECache better trades off performance degradation and power savings and incurs a less than 5% performance degradation in all workloads, except in TH2 (with performance degradation slightly higher than 5%), while providing 14.8% system power savings on average, as shown in Figures 15(a) and 15(b). These results show that our EECache can provide similar power savings to finer-grained approaches with both smaller hardware overheads and less performance degradation.

8. CONCLUSION

This article explores low-cost LLC power management policies for multiprogrammed and multithreaded benchmarks. Based on our extensive experimental analysis, we can conclude that simultaneously exploiting three key metrics, that is, utilization, hotness, and the distribution of dirty cache lines, is necessary to design the power management policies for an energy-efficient LLC. Powering on a fix granularity of cache slices (VHon) can provide higher power savings, while the system that requires higher performance would benefit from the DynOn policy that dynamically powers on a different number of

slices according to the potential hits to the powered-off portions. By migrating data from the hot slices to other active slices, the performance can be improved by eliminating the additional cache misses due to the smaller cache capacity. Our EECache achieves 14.1% energy savings with a less than 2% performance degradation and consumes negligible hardware overhead.

REFERENCES

- B. Ahsan, L. Dreuer, I. Sideris, Y. Sazeides, S. Idgunji, and E. Ozer. 2011. Eliminating energy of same-content-cell-columns of on-chip SRAM arrays. In *Proceedings of the 2011 International Symposium on Low Power Electronics and Design (ISLPED)*. 181–186.
- D. H. Albonesi. 1999. Selective cache ways: On-demand cache resource allocation. In *Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture (MICRO)*. 248–259.
- J. Allred, S. Roy, and K. Chakraborty. 2012. Designing for dark silicon: A methodological perspective on energy efficient systems. In *Proceedings of the 2012 International Symposium on Low Power Electronics and Design (ISLPED)*. 255–260.
- R. Balasubramanian, N. P. Jouppi, and N. Muralimanohar. 2011. *Multi-Core Cache Hierarchies*. Morgan and Claypool.
- A. Bardine, M. Comparetti, P. Foglia, G. Gabrielli, C. A. Prete, and Per Stenström. 2008. Leveraging data promotion for low power D-NUCA caches. In *Proceedings of the 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools (DSD)*. 307–316.
- A. Bardine, M. Comparetti, P. Foglia, G. Gabrielli, and C. A. Prete. 2010. Way adaptable D-NUCA caches. *Int. J. High Perform. Syst. Archit.* 2, 3/4 (Aug. 2010), 215–228.
- A. Bardine, M. Comparetti, P. Foglia, and C. A. Prete. 2014. Evaluation of leakage reduction alternatives for deep submicron dynamic nonuniform cache architecture caches. *IEEE Trans. Very Large Scale Integr. Syst.* 22, 1 (Jan. 2014), 185–190.
- A. Basu, D. R. Hower, M. D. Hill, and M. M. Swift. 2013. FreshCache: Statically and dynamically exploiting dataless ways. In *Proceedings of the 2013 IEEE 31st International Conference on Computer Design (ICCD)*. 286–293.
- N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sadashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. 2011. The gem5 simulator. *SIGARCH Comput. Archit. News* 39, 2 (Aug. 2011), 1–7.
- K. Chandrasekar, C. Weis, Y. Li, S. Goossens, M. Jung, O. Naji, B. Akesson, N. Wehn, and K. Goossens. 2014. DRAMPower: Open-Source DRAM Power and Energy Estimation Tool. Retrieved from <http://www.drampower.info/>.
- Y.-T. Chen, J. Cong, H. Huang, B. Liu, C. Liu, M. Potkonjak, and G. Reinman. 2012. Dynamically reconfigurable hybrid cache: An energy-efficient last-level cache design. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 45–50.
- H.-Y. Cheng, M. Poremba, N. Shahidi, I. Stalev, M. J. Irwin, M. Kandemir, J. Sampson, and Y. Xie. 2014. EECache: Exploiting design choices in energy-efficient last-level caches for chip multiprocessors. In *Proceedings of the 2014 International Symposium on Low Power Electronics and Design (ISLPED)*. 303–306.
- J. Cong, K. Gururaj, H. Huang, C. Liu, G. Reinman, and Yi Zou. 2011. An energy-efficient adaptive hybrid cache. In *Proceedings of the 2011 International Symposium on Low Power Electronics and Design (ISLPED)*. 67–72.
- H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. 2011. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA)*. 365–376.
- K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge. 2002. Drowsy caches: Simple techniques for reducing leakage power. In *Proceedings of the 29th Annual International Symposium on Computer Architecture (ISCA)*. 148–157.
- P. Foglia and M. Comparetti. 2014. A workload independent energy reduction strategy for D-NUCA caches. *J. Supercomput.* 68, 1 (April 2014), 157–182.
- M. Gebhart, J. Hestness, E. Fatehi, P. Gratz, and S. W. Keckler. 2009. *Running PARSEC 2.1 on M5*. Technical Report. The University of Texas at Austin, Department of Computer Science.
- V. George, S. Jahagirdar, C. Tong, K. Smits, S. Damaraju, S. Siers, V. Naydenov, T. Khondker, S. Sarkar, and P. Singh. 2007. Penryn: 45-nm next generation Intel core 2 processor. In *Proceedings of the 2007 IEEE Asian Solid-State Circuits Conference (ASSCC)*. 14–17.

- H. R. Ghasemi, S. C. Draper, and N. S. Kim. 2011. Low-voltage on-chip cache architecture using heterogeneous cell sizes for high-performance processors. In *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture (HPCA)*. 38–49.
- M. Ghosh, E. Ozer, S. Ford, S. Biles, and H.-H. S. Lee. 2009. Way guard: A segmented counting bloom filter approach to reducing energy for set-associative caches. In *Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*. 165–170.
- N. Goulding-Hotta, J. Sampson, Qiaoshi Zheng, V. Bhatt, J. Auricchio, S. Swanson, and M. B. Taylor. 2012. GreenDroid: An architecture for the Dark Silicon Age. In *Proceedings of the 17th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 100–105.
- J. Hu, J. K. John, and S. Wang. 2008. Thermal-aware subarrayed data cache microarchitectures. *Int. J. Intelligent Control Syst.* 13, 4 (Dec. 2008), 251–263.
- A. Jadidi, M. Arjomand, and H. Sarbazi-Azad. 2011. High-endurance and performance-efficient design of hybrid cache architectures through adaptive line replacement. In *Proceedings of the 2011 International Symposium on Low Power Electronics and Design (ISLPED)*. 79–84.
- J. K. John, J. S. Hu, and S. G. Ziavras. 2005. Optimizing the thermal behavior of subarrayed data caches. In *Proceedings of the 2005 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD)*. 625–630.
- D. Kadjjo, H. Kim, P. Gratz, J. Hu, and R. Ayoub. 2013. Power gating with block migration in chip-multiprocessor last-level caches. In *Proceedings of the 2013 IEEE International Conference on Computer Design (ICCD)*. 93–99.
- S. Kaxiras, Z. Hu, and M. Martonosi. 2001. Cache decay: Exploiting generational behavior to reduce cache leakage power. In *Proceedings of the 28th Annual International Symposium on Computer Architecture (ISCA)*. 240–251.
- C. Kim, D. Burger, and S.W. Keckler. 2003. Nonuniform cache architectures for wire-delay dominated on-chip caches. *IEEE Micro* 23, 6 (Nov. 2003), 99–107.
- H. Kim, J.-H. Ahn, and J. Kim. 2010. Replication-aware leakage management in chip multi-processors with private L2 caches. In *Proceedings of the 2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*. 135–140.
- J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. C. Hoe. 2007. Multi-bit error tolerant caches using two-dimensional error coding. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 197–209.
- N. A. Kurd, S. Bhamidipati, C. Mozak, J. L. Miller, T. M. Wilson, M. Nemani, and M. Chowdhury. 2010. Westmere: A family of 32nm IA processors. In *Proceedings of the 2010 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 96–97.
- S. Li, J.-H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 469–480.
- S. Mittal, Z. Zhang, and J. S. Vetter. 2013. FlexiWay: A cache energy saving technique using fine-grained cache reconfiguration. In *Proceedings of the 2013 IEEE International Conference on Computer Design (ICCD)*. 100–107.
- N. Muralimanohar, R. Balasubramonia, and N. P. Jouppi. 2009. *CACI 6.0: A Tool to Model Large Caches*. Technical Report. HP Lab.
- S. Naffziger, B. Stackhouse, T. Grutkowski, D. Josephson, J. Desai, E. Alon, and M. Horowitz. 2006. The implementation of a 2-core, multi-threaded itanium family processor. *IEEE J. Solid-State Circuits* 41, 1, 197–209.
- H. Park, S. Yoo, and S. Lee. 2011. A novel tag access scheme for low power L2 cache. In *Proceedings of the 2011 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1–6.
- H. Park, S. Yoo, and S. Lee. 2012. A multistep tag comparison method for a low-power L2 cache. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 31, 4, 559–572.
- M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar. 2000. Gated-Vdd: A circuit technique to reduce leakage in deep-submicron cache memories. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design (ISLPED)*. 90–95.
- M. K. Qureshi, D. N. Lynch, O. Mutlu, and Y. N. Patt. 2006. A case for MLP-aware cache replacement. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture (ISCA)*. 167–178.
- S. Rusu, S. Tam, H. Muljono, D. Ayers, and J. Chang. 2006. A dual-core multi-threaded Xeon processor with 16MB L3 cache. In *Proceedings of the 2006 IEEE International Solid-State Circuits Conference - Digest of Technical Papers (ISSCC)*. 315–324.

- S. Rusu, S. Tam, H. Muljono, J. Stinson, D. Ayers, J. Chang, R. Varada, M. Ratta, and S. Kottapalli. 2009. A 45nm 8-core enterprise Xeon processor. In *Proceedings of the 2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers (ISSCC)*. 56–57.
- K. T. Sundararajan, V. Porpodas, T. M. Jones, N. P. Topham, and B. Franke. 2012. Cooperative partitioning: Energy-efficient cache partitioning for high-performance CMPs. In *Proceedings of the 2012 IEEE 18th International Symposium on High Performance Computer Architecture (HPCA)*. 1–12.
- K. Swaminathan, E. Kultursay, V. Saripalli, V. Narayanan, and M. Kandemir. 2012. Design space exploration of workload-specific last-level caches. In *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*. 243–248.
- M. B. Taylor. 2012. Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse. In *Proceedings of the 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1131–1136.
- S. Thoziyoor, N. Muralimanohar, and N. P. Jouppi. 2007. *CACTI 5.0*. Technical Report. HP Lab.
- H.-J. Tsai, C.-C. Chen, K.-H. Yang, T.-C. Yang, L.-Y. Huang, C.-H. Chung, M.-F. Chang, and T.-F. Chen. 2014. Leveraging data lifetime for energy-aware last level non-volatile SRAM caches using redundant store elimination. In *Proceedings of the 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6.
- G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M. B. Taylor. 2010. Conservation cores: Reducing the energy of mature computations. In *Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 205–218.
- Y. Wang, S. Roy, and N. Ranganathan. 2012. Run-time power-gating in caches of GPUs for leakage energy savings. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*. 300–303.
- D. Wendel, R. Kalla, R. Cargoni, J. Clables, J. Friedrich, R. Frech, J. Kahle, B. Sinharoy, W. Starke, S. Taylor, S. Weitzel, S. G. Chu, S. Islam, and V. Zyuban. 2010. The implementation of POWER7: A highly parallel and scalable multi-core high-end server processor. In *Proceedings of the 2010 IEEE International Solid-State Circuits Conference - Digest of Technical Papers (ISSCC)*. 102–103.
- C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar, and S.-L. Lu. 2010. Reducing cache power with low-cost, multi-bit error-correcting codes. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA)*. 83–93.
- S.-H. Yang, M. D. Powell, B. Falsafi, and T. N. Vijaykumar. 2002. Exploiting choice in resizable cache design to optimize deep-submicron processor energy-delay. In *Proceedings of the 8th International Symposium on High Performance Computer Architecture (HPCA)*. 151–161.

Received October 2014; revised January 2015; accepted April 2015