SBAC: A Statistics based Cache Bypassing Method for Asymmetric-access Caches

Chao Zhang[†], Guangyu Sun[†], Peng Li[‡], Tao Wang[†], Dimin Niu[§] and Yiran Chen[£]
[†]Center for Energy-Efficient Computing and Applications, EECS, Peking University, Beijing, 100871, China
[‡]Computer Science Department, University of California, Los Angeles, CA, 90095, USA
[§]Dept. of Computer Science and Engineering, Pennsylvania State University, University Park, PA, 16802, USA
[£]Dept. of Electrical & Computer Engineering, University of Pittsburgh, Pittsburgh, PA, 15261, USA
[†]{zhang.chao, gsun, wangtao}@pku.edu.cn, [‡]pengli@cs.ucla.edu, [§]dun118@cse.psu.edu, [£]yic52@pitt.edu

ABSTRACT

Asymmetric-access caches with emerging technologies, such as STT-RAM and RRAM, have become very competitive designs recently. Since the write operations consume more time and energy than read ones, data should bypass an asymmetric-access cache unless the locality can justify the data allocation. However, the asymmetric-access property is not well addressed in prior bypassing approaches, which are not energy efficient and induce non-trivial operation overhead. To overcome these problems, we propose a cache bypassing method, SBAC, based on data locality statistics of the whole cache rather than a single cache line's signature. We observe that the decision-making of SBAC is highly accurate and the optimization technique for SBAC works efficiently for multiple applications running concurrently. Experiments show that SBAC cuts down overall energy consumption by 22.3%, and reduces execution time by 8.3%. Compared to prior approaches, the design overhead of SBAC is trivial.

Categories and Subject Descriptors

B.3.2 [**Design Styles**]: Cache memories; D.4.2 [**Storage Management**]: Allocation/deallocation strategies

Keywords

Statistics; Bypass; Asymmetric-access Cache; Data Reuse Count

1. INTRODUCTION

Non-volatile memories (NVMs), such as spin-transfer torque random access memory (STT-RAM) and resistive random access memory (ReRAM), have been extensively studied to replace SRAM and embedded DRAM (eDRAM) as on-chip caches [6, 21]. Compared to traditional memory technologies, they have advantages of high storage density, low standby power consumption, good scalability, and immunity to particle based soft errors. Prior research has shown that these emerging memories can be employed as L2 and L3 caches to improve performance, reduce power consumption, and even enhance reliability against soft errors [14, 7, 17, 12].

The cache designs based on these emerging memories are normally called ***asymmetric-access caches***. It means that the read and write operations to these memories could be based on different mechanism and demonstrate different access latencies, energy consumptions, and even reliability. In most NVM techniques nowadays, the write latency and energy consumption can be several times larger than those of read. Thus, asymmetry should also be considered in architecture designs.

Prior research has demonstrated that cache bypassing is an efficient technique to mitigate cache contamination problem by selectively allocating data into a cache. There has been extensive research about bypassing techniques for traditional symmetric-access caches [8, 4, 10, 5, 20]. Prior approaches, however, cannot work efficiently with asymmetric-access cache. High overhead of the write operation is left out of consideration, leading to incorrect bypassing decisions. Moreover, in prior approaches, the bypassing decision is cache-line oriented, which means that the access history of every cache line should be tracked. It induces non-trivial design and run-time operation overhead. In addition, some bypassing techniques are designed for specific cache configurations (e.g. exclusive LLC only).

Extensive research has been proposed to mitigate write issues of asymmetric-cache. For example, write halt and PreSET techniques are proposed to hide long write latency of these asymmetric-access caches or main memory [15, 16, 13]. The hybrid cache architecture is explored by allocating frequently updated data to the symmetric-access cache (e.g. SRAM) [15, 19]. The replacement policy can also be tailored [22] by evicting cache lines with less updated bits.

In this work, we propose a statistics based data bypassing method, SBAC, for asymmetric-access caches. The asymmetric cost of read and write operations are well addressed to achieve a proper bypassing decision. Moreover, SBAC makes bypassing decision based on statistical behavior of data in the whole cache, instead of a specific data-block. Consequently, both design and run-time overhead is significantly reduced. More importantly, the bypassing decision-making can achieve high accuracy because the statistical behavior of data is stable and predictable for many applications (details are discussed in Section 2). The results show that our method induces trivial design overhead and can achieve better performance compared to prior approaches. The contribution of this work is summarized as follows:

- We provide theoretical analysis of cost (latency or energy consumption) for allocating or bypassing data into an asymmetric-access cache.
- Based on the theoretical principle, we propose a cache bypassing method, SBAC.
- A run-time bypassing prediction technique is introduced to dynamically adjust bypassing policies.
- We further propose core-based bypassing technique to improve efficiency of SBAC in case that data with different localities are mixed together.

The rest of this paper is organized as follows. The theoretical analysis of statistics based cache bypassing is introduced in Section 2. The architecture and operation flow of SBAC and core-based bypassing techniques are proposed in Section 3.

ISLPED'14, August 11–13, 2014, La Jolla, CA, USA.

http://dx.doi.org/10.1145/2627369.2627611.

The experimental results and discussions are presented in Section 4, followed by conclusions in the last section.

2. THEORY BASIS

In this section, we introduce terminologies and definitions used in theoretical derivation, followed by theoretically exploration of the relationship between data bypassing and data locality.

2.1 Terminologies and Definitions

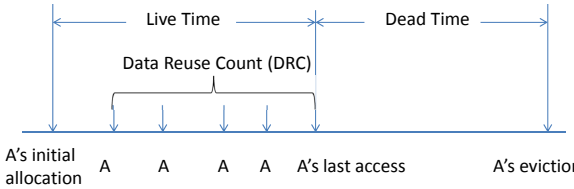

Figure 1: Illustration of data A and related terms [10, 5].

The terminologies used in this work are similar to those in prior literature [10, 5] and are illustrated in Figure 1. As shown in the figure, data A is brought into a cache line by either a read access or a prefetching operation. The life time of A in the cache is composed of live time (from allocation time to the last use) and dead time (from the last use to its eviction). The total number of accesses (hits) to data after the allocation is called ***data reuse count*** (DRC). The cache line A in Figure 1 has a reuse count of five. The first allocation is called ***initial placement***. The data having no live time ($DRC = 0$) is normally called ***instant dead block***.

$$P_i = \frac{N_{DRC=i}}{\sum_{j=0}^{\infty} N_{DRC=j}} \tag{1}$$

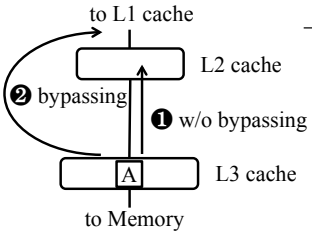

Figure 2: SBAC for loading data.

Term	Definition
DRC	Data Reuse Count
P_i	Probability of $DRC = i$
R_2	Read energy of L2 cache
$R_{2_{tag}}$	Energy of reading L2 tag
$R_{2_{data}}$	Energy of reading L2 data
W_2	Write energy of L2 cache
R_3	Read energy of L3 cache
W_3	Write energy of L3 cache
d	Bypassing depth
λ	Bypassing feature
SI	Sample Interval of DRC

Table 1: Terminologies and definitions.

With DRCs for massive data, we introduce the definition of DRC probability. Let $N_{DRC=i}$ denotes the number of data that have their DRCs equal to i. Then, a DRC probability P_i is calculated in Equation (1). Other definitions and parameters of read and write operations to L2 and L3 caches used in this case study are listed in Table 1.

2.2 Theoretical Energy for Bypassing

We first introduce a case study on data loading. Our goal in this case is ***to reduce cache access energy consumption***. In order to simplify the discussion, we make some assumption. First, there are only read operations to the L2 cache. Second, L3 cache is large enough to allocate the working set. Third, the cache is non-inclusive, so the coherence of data is still kept. As shown in Figure 2, we focus on the case of loading data from L3 to L2. If data loaded from the L3 bypass the L2, they are loaded to L1 directly, as illustrated with path ❷. Otherwise, data will be loaded into L2 normally, shown with path ❶.

We derive the theoretical energy consumption as follows. Initially, the data A is allocated at the L3 only. When the processing core issues a request to access the data A, it generates cache miss at both L1 and L2 and finally receives a cache hit in the L3. If the data are loaded into the L2 without bypassing, the total access energy to L2 can be calculated in Equation (2).

$$E_{w/o\_bypass} = R_3 + W_2 + (DRC + 1) \times R_2 \tag{2}$$

From left to right, the terms on the right side of Equation (2) represent the energy of reading data from L3, writing data to L2, sending data from L2 to L1 after initial placement, and revisiting data for DRC times. If data A bypasses the L2, the total cost will be changed to that in Equation (3).

$$E_{bypass} = (DRC + 1) \times (R_{2_{tag}} + R_3) \tag{3}$$

It means that, for each data access, energy is consumed to detect a cache miss in L2 ($R_{2_{tag}}$) and load data from L3 (R_3). Obviously, we can reduce access energy with cache bypassing only when we have $E_{w/o\_bypass} > E_{bypass}$. Thus, we can obtain Equation (4) as the condition to enable cache bypassing. It means that the DRC should be large enough to ensure the benefits of data reuse, and amortize the overhead of writing data into the L2.

$$DRC < \frac{W_2 + R_2 - R_{2_{tag}}}{R_3 + R_{2_{tag}} - R_2} = \frac{W_2 + R_{2_{data}}}{R_3 - R_{2_{data}}} \tag{4}$$

To bypass or not to bypass, that is the question. For SRAM/eDRAM caches, W_2 is similar to R_2, which is several times smaller than R_3. Cache can benefit from the data allocation whenever there is at least once reuse of the data in L2. For the asymmetric-access cache, however, the W_2 can be comparable to R_3. Thus, a higher DRC is expected to justify the data allocation. In order to achieve lowest access energy, data with DRC less than $\lceil \frac{W_2+R_{2_{data}}}{R_3-R_{2_{data}}} \rceil$ should bypass L2.

In order to demonstrate the impact of read-write asymmetry, we compare the conditions of cache bypassing for SRAM and STT-RAM caches. Table 2 shows typical energy consumption numbers of caches based on SRAM and STT-RAM. For symmetric caches, loading data into L2 is more energy-efficient when DRC is higher than one. While in asymmetric caches, only very frequently accessed data with DRC higher than six should be loaded into L2.

2.3 Theory Basis of SBAC

In practice, it is difficult to exactly know the DRCs of all the data in cache before they all die. However, it is possible to filter out the data with specific DRC with a simple bypassing method. For example, we can assume the average DRC for unfiltered data is smaller than one, and make all initial placements bypass the L2 cache, so only the data with at least one reuse count can enter L2 cache. Thus, the key is to ensure the benefits from dead blocks bypassing can amortize the bypassing of high DRC data.

The theoretical condition of employing bypassing can be derived based on the probabilities of DRCs. Assume that the probability distribution of DRC in L2 is represented by $\{P_0, P_1, P_2, \ldots\}$ ($\sum_{i=0}^{\infty} P_i = 1$). Without bypassing technique, the average access energy of these data is noted as $\bar{E}_{w/o\_bypass}$. If initial placements of whole data bypass the L2 cache, the cache access energy consumption is noted as $\bar{E}_{bypass}$.

$$\bar{E}_{w/o\_bypass} = \sum_{i=0}^{\infty} \{P_i \times [R_3 + W_2 + (i+1) \times R_2]\} \tag{5}$$

$$\bar{E}_{bypass} = P_0 \times (R_3 + R_{2_{tag}}) + \sum_{i=1}^{\infty} \left\{P_i \times \left[2 \times R_3 + R_{2_{tag}} + W_2 + i \times R_2\right]\right\} \tag{6}$$

With these two equations, it is easy to understand that such an "initial placements" bypassing can only reduce average access energy when $E_{bypass} < E_{w/o\_bypass}$. After substituting Equation (5) and (6) into it, we obtain the condition to trigger an "initial placement" bypassing, described as an Equation (7).

$$P_0 > \frac{R_3 + R_{2_{tag}} - R_2}{W_2 + R_3} \tag{7}$$

Cache Type	SRAM L2 STT-RAM L3	STT-RAM L2 STT-RAM L3
$R_{2_{data}}(nJ)$	0.066	0.127
$W_2(nJ)$	0.051	0.603
$R_3(nJ)$	0.246	0.246
$\lceil \frac{W_2+R_{2_{data}}}{R_3-R_{2_{data}}} \rceil$	1	6

Table 2: Typical energy numbers for 2MB SRAM and STT-RAM caches, and 8MB STT-RAM cache (Technology node: 45nm).

It is interesting that the balance point where the benefits can amortize the overhead is only determined by P_0, which is the DRC probability of instant dead blocks. This is the reason why we call our technique SBAC as a statistics based cache bypassing method.

In order to have a quantitative analysis, we calculate the bypass condition for symmetric- and asymmetric-access caches, respectively. Cache bypassing can gain benefits when $P_0 > 62.8\%$ for a symmetric-access cache. For an asymmetric-access cache, however, bypassing condition is satisfied with a significant lower value of $P_0 > 15.5\%$. The parameters we used are listed in Table 2.

2.4 Bypassing Depth

After the "initial placement" bypassing is applied, the original data with once reuse count becomes instant dead block since their first loads are filtered. Thus, it is reasonable to make these new instant dead blocks bypass L2 to further reduce access energy consumption. In other words, bypass the data with $DRC < 2$. Thus, we introduce the definition of ***bypassing depth***, which means that data with DRC less than bypassing depth should bypass the cache. For example, when the bypassing depth is set to "1", only initial placements are bypassed. The calculation of theoretical bypassing depth is discussed as follows.

Similar to the derivation of "initial placement" bypassing decision, we can calculate the bypassing condition with "bypassing depth = 2" as in Equation (8)

$$\frac{P_1}{1 - P_0} > \frac{R_3 + R_{2_{tag}} - R_2}{W_2 + R_3} \tag{8}$$

And we can further calculate condition for any bypassing depth d as in the following Equation:

$$\frac{P_{d-1}}{1 - \sum_{j=0}^{d-2} P_j} > \frac{R_3 + R_{2_{tag}} - R_2}{W_2 + R_3} \tag{9}$$

In this work, the $\lambda = \frac{R_3+R_{2_{tag}}-R_2}{W_2+R_3}$ is called ***bypassing feature*** of the system, which is the intrinsic cache attribute. The high write energy of asymmetric-access caches results in a small bypassing feature, making bypass more attractive to reduce energy consumption.

3. DESIGN OF SBAC

3.1 Overview

We still use the case of loading data from L3 cache to L2 cache to describe the architecture design of SBAC. As a pivot to select proper bypassing decisions, extra components are needed to monitor and predict the distribution of DRC for data in the L2 cache. As shown in Figure 3, one extra bit is added to each cache line in L2, and two bits are added to each cache line of L3 for this purpose. In addition, they are also used to decide whether cache bypassing is needed. The extra function between L2 and L3 is called bypassing decision block **(BDB)**. BDB monitors cache line transferring on the data bus. It can track information of the DRC sent with data so that probability distribution of DRC is calculated.

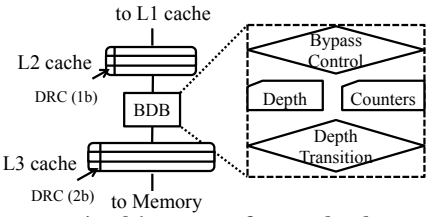

Figure 3: Architecture for cache bypassing.

A BDB includes three global DRC counters. Three DRC counters are denoted as $N_{\geq d-1}$, $N_{\geq d}$, and $N_{\geq d+1}$. They are used to count the number of DRC greater than $d-1$, d, and $d+1$, respectively. With these DRC counters, we can rewrite conditions in Equation (9) with Equation (10). Read/Write energy numbers are used to calculate the λ.

The bypassing control logic can make decision for data transferring on the bus. A cache block will bypass L2 cache if the DRC bit in the L3 cache is smaller than the bypass depth. The bypass depth transition logic is employed to calculate runtime bypassing depth. The bypass depth will be increased by one when Equation (10) is satisfied, and decreased by one when Equation (11) is satisfied.

$$\frac{N_{\geq d} - N_{\geq d+1}}{N_{\geq d}} > \lambda \tag{10}$$

$$\frac{N_{\geq d-1} - N_{\geq d}}{N_{\geq d-1}} < \lambda \tag{11}$$

3.2 Operation Flow with Cache Bypassing

Having the SBAC architecture, we describe the flow for different cache operations with an example in Figure 4. As shown in the figure, L1, L2, and L3 caches are illustrated with one, two, and four cache lines. The three DRC counters of BDB are also shown in the figure. There is one DRC bit for each cache line in L2 and two bits for each cache line in L3. The bypassing depth d is set to 2 in this example. The detailed operation flow is described as follows.

- Step (a): In the initial state, all three DRC counters are initialized as zero. The DRC bits of each line are also cleared as zero. We assume there are some initial data stored in cache lines.
- Step (b): L1 cache requests data C, since the DRC bit of data C in L3 cache is equal to zero, data C is bypassed to L1 cache directly because $DRC = 0 < d = 2$. At the same time, the DRC bit of data C in L3 cache is increased by one.
- Step (c): Similarly, when L1 cache requests data D, it is also moved from L3 to L1 directly for the same reason.
- Step (d): When L1 cache requests data C again, data C is bypassed again because we still have $DRC = 1 < d = 2$. Then, DRC of data C in L3 cache is increased to 2. At the same time, the first counter in BDB is increased by one because it counts the number of data with $DRC \geq d - 1 = 1$.
- Step (e): Similarly, when L1 cache requests data D again, it is bypassed again. And the first DRC counter of BDB is increased by one.

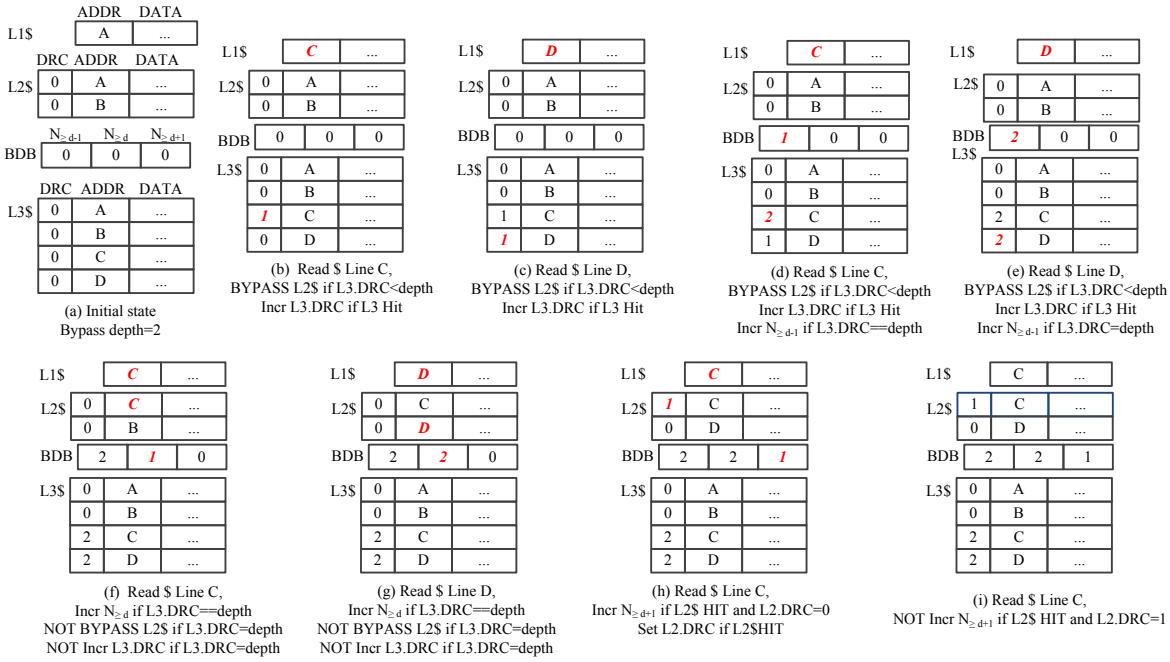

Figure 4: An example of cache bypassing flow.

- Step (f): When L1 cache requests data C for the third time, data C is finally loaded to L2 cache because we have $DRC = d$ now. At the same time, the second counter in BDB is increased by one. Note that the DRC bits of data C in L3 cache are saturated now. They are only reset to zero when data C are evicted from L3 cache.
- Step (g): Similarly, data D is also loaded to L2 cache for the third request, and the second counter in BDB is increased by one.
- Step (h): When data C is first hit in L2 cache, the third counter in BDB is increased by one because C is requested for $d + 1 = 3$ times in total. At the same time, its DRC bit in L2 is set to one.
- Step (i): When data C gets hit again with DRC bit equal to one, the third counter in BDB remains the same.

3.3 Sensitivity Control

Since the probability distribution of DRC varies during run-time execution, the bypassing depth should also be updated periodically to reflect the distribution. The length of each period, in terms of cache accesses, is called ***sampling interval (SI)*** in this work. At the end of a sampling interval, the BDB counters are used to calculate the current probability distribution of DRC. The distribution is used to predict the bypassing depth for the next interval.

The choice of sampling interval has an impact on the prediction accuracy of bypassing depth. Since the bypassing depth is based on DRC, we use the amount of cache access to determine a SI. If the SI is too short, the poor sampled statistics cannot represent the probability distribution of DRC. On the other hand, if the SI is too long, it may not capture the changes of DRC distribution so that the efficiency of SBAC is degraded. In addition, the size of counters in BDB is also related to SI.

Experimental results show that the optimal SI varies in the range of $10k \sim 100k$ for different workloads. Thus, we propose an algorithm to dynamically adjust SI for different data patterns. The algorithm is described as follows.

- SI is initialed as 2^{14}, which is the lower bound of SI.
- After each SI, if the bypassing depth is not changed, SI is increased by $2\times$.
- After each SI, if the bypassing depth is changed, SI is decreased by $2\times$.
- The higher bound of SI is set to 2^{20}. Thus, a 20-bit counter is needed.

3.4 SBAC Extension for Other Scenarios

Extenstion for Performance Optimization. To apply SBAC for cache performance optimization, we need to replace energy numbers in equation (1) - (9) with proper access latency numbers. Different from energy consumption, it is inaccurate to add the latency of a write operation directly to the total execution time. Instead, we need to estimate the time that L2 cache is blocked due to loading data from L3. The blocking time is related to cache access intensity. Previous research [15, 19, 22] pointed out that the blocking time varies from $0\times$ to $0.6 \times$ *write latency*. One solution to this problem is to calculate average run-time blocking time by monitoring the waiting time of read operations in the miss status holding registers (MSHRs).

Extension for Multi-core Optimization. For the case that there are multiple worloads running on multiple cores, more BDBs can be added to track DRC distribution of each core separately. The core ID needs to be integrated in the cache tag to identify data from each core. Thus, bypassing decisions may be different for data requested by different cores to improve efficiency of SBAC. Such an extension of SBAC is called "core-based SBAC". Extra design overhead is induced because the number of counters increase proportionally with the number of cores. Note that SBAC can be applied to both shared and private caches. For example, if L2 cache is private for each core and L3 cache is shared, each L2 needs one BDB to connect L3.

4. EXPERIMENTAL EVALUATION

In this section, we provide comprehensive evaluation to demonstrate the efficiency of SBAC for single and multiple applications under both shared and private L2 configurations.

4.1 Experiment Setup

We implement SBAC in a popular full-system simulator *gem5* [2]. It is configured to model a four-core Haswell like CMP. Each core is running at 2GHz frequency. There

are three levels of caches. The IL1/DL1 caches are SRAM based and and the L2 and L3 are configured as asymmetric-access STT-RAM caches. Other details can be found in Table 3. We use cache latency and energy parameters from NVSim [3].

Component	Configuration
Processor	4 cores, 2GHz, 1-way issue
IL1/DL1 SRAM	32/32KB, 2-way, 64B, private, LRU L.P.:47.7mW, R/W Lat.: 2/2cycle, E.:6.2/2.3pJ
L2 STT-RAM	4 × 256KB, 8-way, 64B, LRU, L.P.:428mW R/W Lat.: 6/36cycle, E: 0.135/0.603nJ
L3 STT-RAM	8MB, 16-way, 64B, share, LRU, L.P.:1851mW R/W Lat.: 25/60cycle, E: 0.246/0.698nJ
Memory	8GB, DDR3, 1600MHz, 120cycle, 12.8GB/s.

Table 3: Detailed simulation setup.

Both single and multiple applications workloads are evaluated. In order to provide a comprehensive evaluation with diversified distributions of DRC, we examine different code segments in both single and randomly mixed multi-programmed benchmarks. Both private and shared L2 configurations are used for experiments of mutli-programmed workloads. For the single application case, only the private cache with one core running is evaluated. The simulator captures all data operations such as loads, stores, and prefetching requests. The one block lookahead (OBL) approach is employed for prefetching in evaluation. All benchmarks come from SPEC CPU 2006. We fast forward one billion instructions at beginning, and execute ten billion instructions of a single benchmark. Then we construct the multi-program workloads by mixing the fast forwarded single programs. Energy consumption includes leakage and dynamic power of entire cache hierarchy, based on operation statistics.

The labels used in the rest of this section are explained: (1) Baseline: baseline case without cache bypassing; (2) SBAC: case using SBAC; (3) SBAC-C: case using core-based SBAC; (4) Shared: case with shared L2 configuration; (5) Private: case with private L2 cache.

4.2 DRC Prediction Accuracy

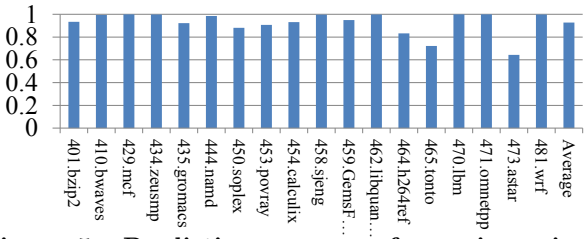

Figure 5: Prediction accuracy for various single-programmed benchmarks.

As shown in Figure 5, a high prediction accuracy of 92% on average is achieved for single program benchmarks. The prediction accuracy is about 86% on average for multiprogrammed applications (not shown due to page limit). Note that a correct bypassing decision may obtained even with a mis-prediction, as long as the bypassing depth is not affected. On the other hand, a correct prediction of DRC distribution may also lead to an incorrect decision of cache bypassing due to inaccurate estimation of read/write energy.

4.3 Evaluation for Single Application

The results of energy consumption are compared in Figure 6. We can find that the reduction of energy is related to the prediction accuracy generally. For some benchmarks, however, the energy reduction is insignificant even with high prediction accuracy (e.g. *GemsFDTD*). The reason is that for some benchmarks the cache bypassing is not triggered for most of execution time. On average the reduction of the total cache energy consumption is about 22.3%.

The results of performance improvement is similar to energy reduction, but less significant. The reason is that the energy consumption of each load operation is reflected in total energy, but the loading time could be hidden by MSHR. On average, the total execution time is reduced by 8.3%. Detailed results are not included due to page limitation.

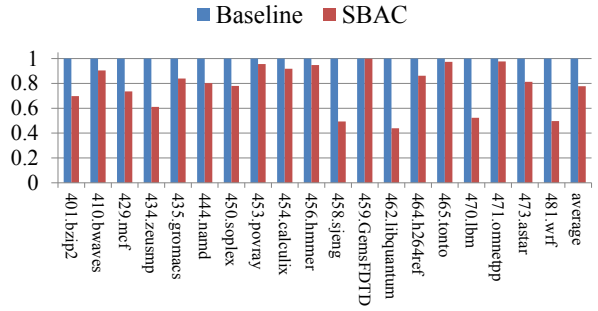

Figure 6: Normalized energy consumption for single applications.

4.4 Evaluation for Multi-programmed Applications

We evaluate energy consumption after applying SBAC to two cache configurations against corresponding baselines without cache bypassing. We show the normalized comparison in Figure 7. The results demonstrate that, for private cache configuration, the energy consumption can be reduced after using SBAC. It is because each workload is bounded to a dedicated core and the DRC distribution is estimated separately. On average, SBAC can reduce energy consumption by 7.5% for private L2 cache, but 3.8% for shared L2 cache configuration. As addressed before, mixing data with different patterns from multiple workloads makes SBAC less efficient. In order to improve SBAC for multi-programmed

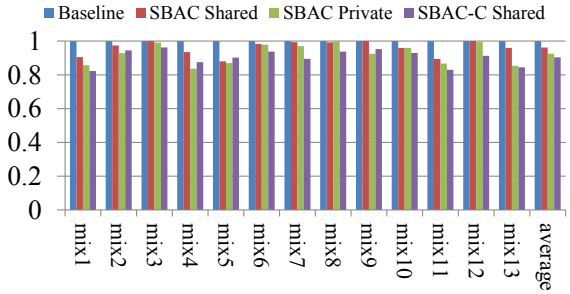

Figure 7: Normalized energy consumption after using SBAC for two cache configurations.

workloads, we propose core-based SBAC for shared L2, listed as the fourth bar in Figure 7. It is easy to find that energy consumption is further reduced after using the technique. Core-based SBAC can further reduce the energy consumption by about 9.9%, because it helps isolate the interference of data amoung different workloads, while shared cache supplies sufficient space. We also evaluate the results of execution time after using SBAC and compare them with the baseline. The results of applying SBAC and SBAC-C on data loaded from L3 to L2 are listed in Figure 8. We can find that the trends of these results are similar to those for energy consumption optimization. On average, the performance is improved by 2.1% and 4.3% for shared and private

Approaches	Multiprogram Support	Storage Overhead (bits)		Area Overhead (μm^2)	Operation Overhead
		per line	global		
DBP [11]	No	-	2M	102.78	2-level table lookup/update
IATAC [1]	No	31	288	293.63	6b comp + 31b update + 16-entry CAM lookup/update
IGDR [18]	No	-	42.5K	4.12×10^4	5 table lookup/update
LvP [10]	No	17	40K	62.27	5b comp + 17b update + 1 table lookup/update
AIP [10]	No	21	40K	30.34	5b comp + 21b update + 1 table lookup/update
DBRB [9]	Yes	-	13.75K	40.28	15b comp + 15b update + 3 table lookup/update
BIA [5]	Yes	3/L2+2/L3	1.8K	192.6	5b update + 16-entry CAM lookup/update
SBAC	**Yes**	**1/L2+2/L3**	**73**	**36.16**	**3b update + 2b comp + 1 counter update**
SBAC-C	Yes	1/L2+2/L3	146	144.64	3b update + 2b comp + 1 counter update

Table 4: Design overhead comparison.

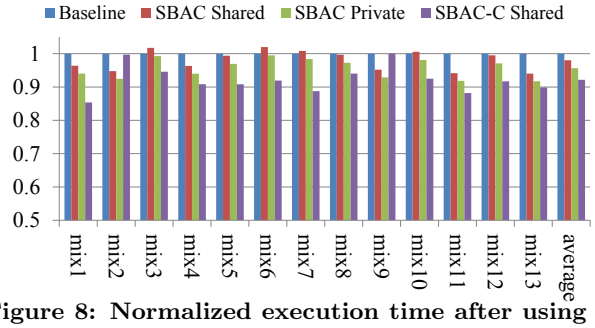

Figure 8: Normalized execution time after using SBAC for two cache configurations.

configured L2 cache. And the core-based SBAC can improve performance by 9.4% for the shared L2 cache.

4.5 Comparison with Other Approaches

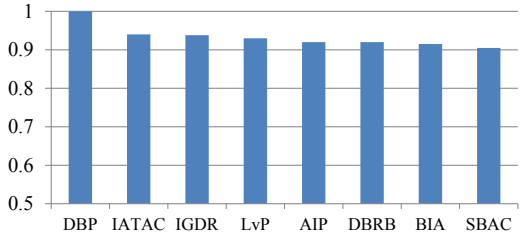

Figure 9: Comparison of performance between our bypassing scheme and other approaches.

We compare normalized average cache access latency between our bypassing scheme and prior approaches for single application, shown in Figure 9. Our bypassing scheme can outperform other approaches in respect of cache access latency. The main reason is that the asymmetry access operations are not considered in prior approaches. Note that we do not provide comparison for cases of energy optimization and multi-programmed application. It is because most prior approaches cannot work with these cases. We also compare design and operation overhead in Table 4. We estimate the design overhead by extra storage (per line and cache), area overhead of control logic, and extra cache operations. Area results are synthesized by Synopsys Design Compiler with TSMC 45nm library. It is easy to find that SBAC costs much less storage, area, and operations.

5. CONCLUSION

Emerging asymmetric-access caches are competitive for design of future cache hierarchy. Traditional cache bypassing techniques are not efficient for these asymmetric-access caches. In this work, we propose the statistics based cache bypassing method named SBAC. With the help of a theoretical model, we analyze the benefits of cache bypassing. Then, proper bypassing decisions are made based on DRC probability. In addition, we propose core-based SBAC to improve working efficiency of SBAC for multi-programmed workloads. Compared with prior approaches, SBAC has the advantages of low design overhead and compatibility for different cache configurations. The experimental results show improvement of cache performance and energy efficiency after using SBAC.

6. ACKNOWLEDGEMENTS

This paper is supported by NSF CNS-1116171, National Natural Science Foundation of China (No.61202072 and No.61103028), and National High-tech R&D Program of China (No.2013AA013201).

7. REFERENCES

[1] J. Abella, A. González, X. Vera, and M. F. P. O'Boyle. Iatac: a smart predictor to turn-off l2 cache lines. *ACM Trans. Archit. Code Optim.*, 2(1):55–77, Mar. 2005.

[2] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug. 2011.

[3] X. Dong, C. Xu, Y. Xie, and N. Jouppi. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(7):994–1007, 2012.

[4] H. Dybdahl and P. Stenström. Enhancing last-level cache performance by block bypassing and early miss determination. ACSAC'06, pages 52–66. Springer-Verlag, 2006.

[5] J. Gaur, M. Chaudhuri, and S. Subramoney. Bypass and insertion algorithms for exclusive last-level caches. ISCA'11, pages 81–92. ACM, 2011.

[6] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, and *et al.* A Novel Non-Volatile Memory With Spin Torque Transfer Magnetization Switching: Spin-RAM. In *Proceedings of IEDM*, pages 459–462, 2005.

[7] A. Jadidi, M. Arjomand, and H. Sarbazi-Azad. High-endurance and performance-efficient design of hybrid cache architectures through adaptive line replacement. In *ISLPED'11*, pages 79–84, 2011.

[8] T. L. Johnson, D. A. Connors, M. C. Merten, and W.-m. W. Hwu. Run-time cache bypassing. *IEEE Trans. Comput.*, 48(12):1338–1354, Dec. 1999.

[9] S. M. Khan, Y. Tian, and D. A. Jimenez. Sampling dead block prediction for last-level caches. MICRO'10, pages 175–186, 2010.

[10] M. Kharbutli and Y. Solihin. Counter-based cache replacement and bypassing algorithms. *IEEE Trans. Comput.*, 57(4):433–447, Apr. 2008.

[11] A.-C. Lai, C. Fide, and B. Falsafi. Dead-block prediction & dead-block correlating prefetchers. ISCA'01, pages 144–154. ACM, 2001.

[12] J. Li, P. Ndai, A. Goel, H. Liu, and K. Roy. An alternate design paradigm for robust spin-torque transfer magnetic ram (stt mram) from circuit/architecture perspective. In *ASP-DAC'09*, pages 841–846, 2009.

[13] M. K. Qureshi, M. M. Franceschini, A. Jagmohan, and L. A. Lastras. Preset: improving performance of phase change memories by exploiting asymmetry in write times. ISCA '12, pages 380–391. IEEE Press, 2012.

[14] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan. Relaxing non-volatility for fast and energy-efficient stt-ram caches. In *HPCA'11*, 2011.

[15] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen. A novel architecture of the 3d stacked mram l2 cache for cmps. In *HPCA'09*, pages 239 –249, feb. 2009.

[16] G. Sun, Y. Zhang, Y. Wang, and Y. Chen. Improving energy efficiency of write-asymmetric memories by log style write. ISLPED '12, pages 173–178. ACM, 2012.

[17] Z. Sun, X. Bi, and H. Li. Process variation aware data management for stt-ram cache design. ISLPED '12, pages 179–184. ACM, 2012.

[18] M. Takagi and K. Hiraki. Inter-reference gap distribution replacement: an improved replacement algorithm for set-associative caches. ICS'04, pages 20–30. ACM, 2004.

[19] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie. Hybrid cache architecture with disparate memory technologies. ISCA'09.

[20] Y. Wu, R. Rakvic, L.-L. Chen, C.-C. Miao, G. Chrysos, and J. Fang. Compiler managed micro-cache bypassing for high performance epic processors. MICRO'02, pages 134 – 145, 2002.

[21] C. Xu, X. Dong, N. P. Jouppi, and Y. Xie. Design implications of memristor-based RRAM cross-point structures.

[22] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. Energy reduction for stt-ram using early write termination. In *ICCAD'09*, pages 264–268, 2009.