J Inf Process Syst, Vol.17, No.1, pp.51~62, February 2021
https://doi.org/10.3745/JIPS.01.0062

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

New Two-Level L1 Data Cache Bypassing Technique for High Performance GPUs

Gwang Bok Kim* and Cheol Hong Kim**

Abstract

On-chip caches of graphics processing units (GPUs) have contributed to improved GPU performance by reducing long memory access latency. However, cache efficiency remains low despite the facts that recent GPUs have considerably mitigated the bottleneck problem of L1 data cache. Although the cache miss rate is a reasonable metric for cache efficiency, it is not necessarily proportional to GPU performance. In this study, we introduce a second key determinant to overcome the problem of predicting the performance gains from L1 data cache based on the assumption that miss rate only is not accurate. The proposed technique estimates the benefits of the cache by measuring the balance between cache efficiency and throughput. The throughput of the cache is predicted based on the warp occupancy information in the warp pool. Then, the warp occupancy is used for a second bypass phase when workloads show an ambiguous miss rate. In our proposed architecture, the L1 data cache is turned off for a long period when the warp occupancy is not high. Our two-level bypassing technique can be applied to recent GPU models and improves the performance by 6% on average compared to the architecture without bypassing. Moreover, it outperforms the conventional bottleneck-based bypassing techniques.

Keywords

Bypassing, Cache, GPU, Miss Rate, Performance

1. Introduction

Massively parallel processing with graphics processing units (GPUs) is used not only for graphics computation but also for general purpose workloads [1]. In general, the GPU employs an on-chip L1 cache to reduce long access latency of off-chip memory. Traditionally, a streaming multiprocessor (SM) is equipped with a small L1 cache. Unfortunately, the GPU cache is used inefficiently compared to the CPU cache due to massive multithreading. Many cache blocks in the GPU cache are evicted before they are re-referenced since the requests from many threads compete for small cache resources. Thus, even with the latest GPU architecture such as Pascal [2], the efficiency of the L1 data cache is still quite low. Therefore, if the L1 data cache can be bypassed dynamically according to the characteristics of applications, it can be an opportunity to improve the performance of the GPU.

In recent GPUs, cache resources such as miss status holding registers (MSHRs) and miss queues are easily saturated by high miss rate and high thread parallelism, resulting in the bottleneck problem of the

Manuscript received September 15, 2020; first revision November 16, 2020; accepted November 18, 2020.

Corresponding Author: Cheol Hong Kim (cheolhong@ssu.ac.kr)
* R&D Center 2, SFA Engineering, Hwaseong, Korea (gwangbokkim@sfa.co.kr)
** School of Computer Science and Engineering, Soongsil University, Seoul, Korea (cheolhong@ssu.ac.kr)

L1 cache. For this reason, many GPU-related studies choose an approach which reduces the parallelism to solve the bottleneck problem that occurs in the L1 data cache, i.e., the number of threads is throttled, or cache is bypassed dynamically [1,3,4]. Recently, the GPU shows a structural change for the cache hierarchy to leverage high thread-level parallelism (TLP). For instance, Volta generation employs advanced cache and more cache resources to prevent severe bottlenecks causing overall GPU performance degradation [5].

In this study, we propose a new bypassing technique utilizing the TLP information to predict the cache usefulness. This study includes the following key contributions. First, we address a new issue concerning the warp occupancy in terms of cache usage. In previous GPU architectures, many studies have sought to throttle the warp or cache access, as high TLP does not always guarantee high performance [6]. However, recent GPU cache alleviates the bottleneck problem by adopting the design that maximizes the throughput. Thus, we demonstrate the relation between the TLP and cache benefits to improve the GPU performance. Second, we propose a new approach to bypass the L1 data cache in the GPU. Our two-level decision method resolves the problem that a single metric has limitation to predict the caching efficiency. The second determinant, TLP, is used to determine the bypass for workloads that show ambiguous cache miss rates in our proposed method.

The remainder of this paper is organized as follows. Section 2 describes related work and the motivation of our proposed method. Section 3 presents our proposed bypassing technique for improving the GPU performance. Section 4 provides our evaluation methodology and results in detail. Finally, Section 5 concludes this paper.

2. Background

2.1 Related Work

Most of the research related to the cache management of the GPU focused on leveraging data locality or alleviating bottlenecks. Previous studies have demonstrated the effectiveness of their methods for the L1 cache since on-chip memory is critical to GPU performance. Most previous studies consider thrashing or the bottleneck of the L1 cache, which are caused by the contention of massive threads. Thus, the number of active warps is throttled or access to the cache is limited in previous research. To reduce intra-warp or inter-warp contention, warp/thread block (TB) level throttling techniques have been proposed [7]. Another approach is cache bypassing techniques, which dynamically determine the access to the on-chip cache based on the information of requests. In GPUs, the data locality cannot be utilized properly because of the thrashing problem that cache data are replaced before the data are reused. Several cache management techniques based on the miss rate for the CPU have been proposed [8]. However, hit-rate-based methods do not always perform well in the GPU [9]. Jia et al. [1] proposed a memory request prioritization buffer (MRPB) to improve the cache efficiency of the GPU. MRPB reorders the requests by placing them into different queues. Memory requests are stored in multiple queues before accessing the L1 cache. In addition, apart from the reordering scheme, a bypassing technique that dynamically bypasses cache access when a stall occurs in the cache has been proposed. This technique improves the GPU performance by combining request reordering and cache bypassing techniques. Sethia et al. [10]

proposed Mascar (memory aware scheduling and cache access re-execution) considering same issue. Mascar reduced the cache thrashing by reordering the pending requests when load/store (LD/ST) unit cannot process a miss request. If the request occurs a miss, it is transmitted to the lower level memory and occupies hardware resources such as the MSHR to manage the missed request. When the LD/ST unit is stalled due to memory saturation, a miss request cannot be processed immediately, in this case, the memory request is moved back to the re-execution queue for later re-access.

Dynamic bypassing schemes for GPUs can be classified into two categories. The first category includes the methods that dynamically perform bypass decisions by predicting the locality of each thread/warp. Koo et al. [11] proposed an access pattern-aware cache management (APCM) technique which bypasses streaming access and protects cache blocks with the intra-warp locality. Because many benchmarks often include streaming data, the APCM can reduce the contention of the L1 data cache by bypassing streaming accesses. The APCM also utilizes the intra-warp locality in the L1 data cache. Zhang et al. [4] proposed warp throttling and bypassing based on cache contention. They used the VTT (victim tag table) to estimate the lost-locality score. Since resource contention impacts the GPU performance considerably, the memory requests are bypassed at L1 data cache after selecting the warp which shows a low lost-locality score. As the thread volume gets larger, the throughput of the memory system increases, however from a certain value, cache sensitive applications experience a sudden decrease in performance. The second category includes workload characteristics-based bypassing approaches as presented in [8]. These techniques maintain bypassing policies for a period of time because the workload characteristics are similarly maintained over a single kernel or long cycles. They use a periodic time window for bypass, but do not unconditionally bypass all memory requests within a bypassing period different to our proposed technique in this paper.

2.2 Motivation

Fig. 1 shows miss rate and warp occupancy across 12 benchmarks in our experiments. Simulated benchmarks show high miss rates on average, even though the advanced GPU cache architecture is applied. If a cache miss occurs, then additional cycles for accessing the lower memory is required, thus degrading the performance.

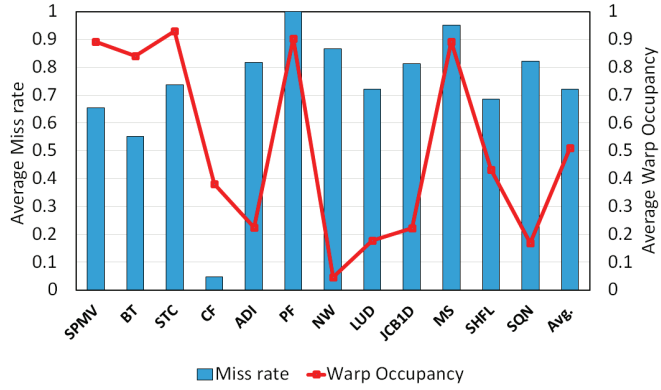

Fig. 1. Miss rates of L1 data cache and warp occupancy.

Fig. 1 also represents the proportion of active concurrent warps. We measured average thread-level parallelism for each benchmark by monitoring the number of active warps. Except for CF, PF, and MS which show either extremely high or low miss rate (4.6%, 99%, and 95%, respectively), most benchmarks exhibit clear relation between warp occupancy and cache usefulness. SPMV, BT, and STC which favor L1 data cache exhibit high TLP (89%, 84%, and 93% warp occupancy, respectively). On the contrary, the other benchmarks which use cache inefficiently represent warp occupancy less than 50%. Therefore, we can know that high TLP is correlated with performance gains from the L1 data cache. For SPMV, BT, and STC benchmarks, the performance of the GPU is degraded when the L1 data cache is turned off, because they show relatively low miss rate. The benefits from the cache can be predicted by cache miss rate since the miss rate reflects the affinity of the workload for the cache. However, even though LUD (72%) and SHFL (68%) show similar miss rate to STC (73%), the performance is improved when the L1 data cache is turned off. Hence, cache miss rate as a single metric is not proper to judge whether the cache is beneficial to workload even though cache miss rate is a reasonable metric to evaluate the cache efficiency. In this paper, we propose a technique which bypasses L1 data cache dynamically by identifying the characteristics of applications.

3. Proposed Two-Level Dynamic L1D Bypassing Scheme

3.1 Two-level Bypassing Method

Even if the data locality is high, the resources are underutilized when the bottleneck problem is severe in the cache of GPUs. Therefore, cache benefits for GPU performance cannot be predicted by cache miss rate only. However, for recent GPU architectures, the pipeline stalls due to the unavailability of resources in the L1 data cache have been reduced significantly. Hence, cache miss rate has become more reliable metric to predict the cache benefits. In this study, we propose a new bypassing technique that adopts two-level decision approach. The first step is to filter the workloads that have relatively clear characteristics about cache friendliness. The high miss rate workloads and considerably low miss rate ones can be easily classified by monitoring the cache miss rate. The efficiency of the L1 data cache is estimated through a sampling method without detecting the individual thread or warp-level requests that have data locality. The workload that has a high cache miss rate close to 100% usually does not obtain the benefits from the cache. Our method easily makes the bypassing decision for these kinds of workloads. When the number of re-referenced blocks is bigger than the number of blocks that are evicted without being re-referenced, allowing L1 data cache access can improve the performance. However, even though the miss rate of L1 data cache is a reasonable metric to present the characteristics of applications, the miss rate only is not proportional to the performance of GPUs.

In the proposed two-level bypassing technique, the miss rate is measured by counting the number of cache accesses and misses during sufficient number of cycles. If the specified sampling period has elapsed, the bypass mode is determined through the two-step mechanism. Our scheme uses two modes that indicate whether to bypass or not at the end of the sampling period. The caching mode (C mode) indicates the state that allows continuous cache accesses, and the bypassing mode (B mode) represents the state that does not allow cache accesses during a long period. Thus, in the case of the B mode, the cache resources as well as

the additional access cycles are not consumed. The proposed technique can alternate between the two modes (C mode and B mode) dynamically. If the degree of data reuse in the L1 data cache is high enough to improve the performance of GPUs, the cache can be beneficial. Therefore, if the monitored miss rate of L1 data cache is lower than the predetermined low threshold ($threshold_L$), then C mode is enabled. By contrast, the B mode is enabled when a workload with higher miss rate than high threshold ($threshold_H$) is detected. Therefore, the first stage of our technique seeks to apply a suitable caching policy for each workload characteristics that have a significantly low or high miss rate.

A second metric, the TLP is used for determining the bypassing when the monitored miss rates is ambiguous (between low threshold and high threshold). If the miss rate does not meet the preceding conditions at the first stage, it is difficult to predict exactly whether the GPU performance benefits from using the L1 data cache. Therefore, our proposed technique tries to estimate the cache benefits by measuring the TLP level together. The performance gains from the on-chip cache can be obtained when the TLP is high, resulting in access latency hiding by multiple threads. At the second stage in the proposed method, the warp occupancy is measured to predict the number of threads accessing the cache. If the warp occupancy is lower than the value of $threshold_W$, it activates B mode to prevent accessing the L1 data cache. Otherwise, the C mode is enabled to turn on the L1 data cache. In GPUs, the warp scheduler at the issue stage selects the warp to execute an instruction. However, if no active warps exist that can be issued, the warp scheduler does not issue any warp. Because the LD/ST unit receives the memory instructions from the issue stage, memory-level parallelism decreases when the number of active warps is small. Therefore, a sufficient quantity of warp pools helps memory-level throughput and utilization of resources on an SM. If a cache does not allow pending miss requests and multithreading, the benefits derived from the L1 cache are mainly influenced by the cache hit rate. However, the GPU cache allows the requests simultaneously and allows massive multithreads. In addition, many GPU applications have memory-intensive workloads, which make GPU cache busy. Our proposed technique encourages high parallelism in the L1 data cache since the cache bottleneck is considerably reduced in the recent GPU architectures such as Pascal and Volta. Our two-level bypassing technique adopts the TLP as the second determinant for bypassing decision, since high TLP can increase the utilization of cache resources. Moreover, high TLP can reduce the overall processing time by overlapping multiple memory requests. Unlike previous architectures that suffer from LD/ST unit stalls, the latest GPU cache architecture provides relatively low penalty of cache contention. If the workload is executed with a large number of concurrent threads, the throughput-oriented GPU cache would be advantageous to hide the penalty. Therefore, we propose the technique to monitor the warp occupancy, which denotes the degree of warp-level parallelism.

3.2 Hardware Implementation

Fig. 2 shows the modified GPU architecture to support our proposed bypassing technique. The warp occupancy, the metric in the second bypassing decision phase, is monitored in the instruction buffer by warp occupancy monitor unit (WOMU). The warp occupancy is the total number of active warps that can be executed. The WOMU proposed in this study counts the number of active warps by looking up the instruction buffer. The instruction buffer stores the information of each warp to execute an instruction. The next instruction of each warp is fetched from the instruction cache using the PC. The instruction

buffer also holds the information indicating the validity of these fetched and decoded instructions. The number of active warps can be checked every cycle through the valid bit of each entry in the instruction buffer. The second determinant, TLP degree is estimated by comparing the counted warp occupancy with predefined value ($threshold_W$). Then, the comparison result of 1 bit is transmitted to the bypassing decision unit (BDU). The miss rate monitor unit (MRMU) records the number of accesses and misses based on the requests to access the L1 data cache. The MRMU calculates the miss rate after the sampling period. Sampling begins when a thread block is allocated to each SM after a new kernel launches on the GPU.

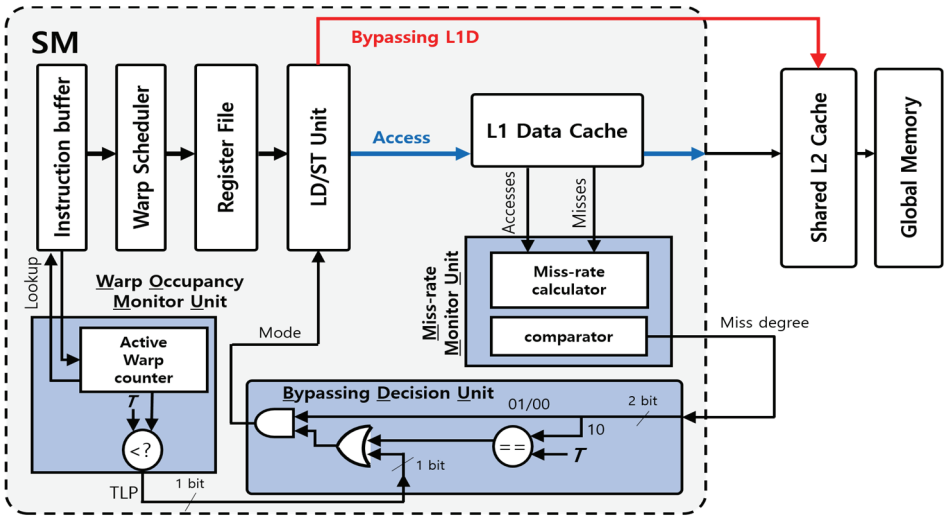

Fig. 2. Hardware modification for the proposed technique.

The proposed technique monitors L1 data cache during a predefined sampling period (i.e., 5K cycles). In this work, after the experiments on various sampling period lengths, we found that the performance result is not so sensitive to the sampling period length. The monitoring techniques for cache behavior have been previously studied related to CPU and GPU architectures [12]. Thus, implementation of miss rate monitoring is not a critical concern in terms of hardware design.

The BDU receives the result from MRMU which compares the miss rate with the predefined $threshold_H$ and $threshold_L$. If a miss rate calculated from the MRMU is less than the $threshold_L$, "0" is passed to the BDU. The BDU determines C mode which allows cache accesses when it receive "0". In the same manner, when the miss rate is higher than $threshold_H$, "1" is passed to the BDU to switch to B mode. The LD/ST unit bypasses the L1 data cache as the BDU is switched to B mode. If the cache miss rate is lower than $threshold_H$ and higher than $threshold_L$, the cache bypass policy cannot be determined immediately. Then, the MRMU sends a value of "2" to the BDU to proceed with the second bypassing decision. Our bypassing technique employs a second decision phase for a workload with an ambiguous miss rate range. The BDU consists of simple hardware logics. First, a signal is sent to the LD/ST unit from the BDU so that the cache can be accessed during the sampling period. When the sampling period ends, the bypass mode is determined using the values received from the MRMU and WOMU. The BDU is responsible for determining whether to send the memory requests to the L1 data

cache or lower-level memory according to the final decision result. Note that the proposed technique maintains cache coherency by bypassing only load instructions. When B mode is determined, store instructions are allowed to access the L1 data cache while load instructions bypass the L1 data cache. In the GPU, the L1 data cache does not allow store instruction to allocate the cache block when a miss occurs. When a write hit happens, the new data is written into the L1 data cache and lower memory based on write-through policy. The proposed two-level bypassing technique requires considerably small hardware overhead, as it consists of only a few comparators and counters. For each SM, 2×13 bits counters are required to store the number of accesses and misses during the sampling period. In addition, only 5 bits for storing warp occupancy is required.

4. Performance Analysis

We simulated our proposed architecture using GPGPU-Sim v4.0 [13], which is a cycle-accurate simulator and is widely used in the research on GPU architecture. We use the GPGPU-Sim with default configuration based on the NVIDIA GTX 1080 state-of-the-art architecture [14]. We set the hit latency of L1 data cache with 82 cycles which is measured in our experiments, and it is same value with previous studies [15,16]. Table 1 presents main parameters of the simulated system. We chose 12 GPU benchmarks from Rodinia, Polybench, CUDA SDK 8.0 and Tango [17] to evaluate the cache efficiency and GPU performance.

Table 1. System configuration

Parameter	Value
SIMT Core/Width	56 cores/32 threads
L1 Inst. Cache	4 kB, 4-way, 128 B line
L1 Data Cache	Sector cache 24 kB, 48-way, 128 B line, Allocation-on-fill, MSHR entry: 16, Miss queue entry: 32, Hit latency: 82 cycles
Scheduler	GTO warp scheduling, RR CTA scheduling
Interconnect	Fly, 32 B channel width
Clock	1417 (Core) : 1417 (Interconnection) : 1417 (L2) : 2500 (DRAM) (MHz)

Fig. 3 shows the performance results according to miss rate threshold-based bypassing. Each bar shows the performance result with different miss rate threshold k. For example, if threshold k is set as 0.6, then the L1 data cache is bypassed when the workload shows a higher cache miss rate than 0.6. We had simulations varying the value of k from 0.6 to 0.9. The case in which the L1 data cache is turned off is also included. As shown in this graph, some benchmarks show an IPC (instructions per cycle) improvement according to different values of k. In general, the benchmarks which are favorable to cache, SPMV, BT, STC, and CF show better performance based on relatively higher values of k because these benchmarks have better data locality than other benchmarks. In contrast, relatively lower threshold k tends to show better performance for benchmarks that prefers B mode (e.g., ADI, PF, NW, LUD, JCB1D, MS, SHFL, and SQN). Each benchmark provides a different optimal threshold value for

peak IPC when miss rate-based bypassing with fixed threshold is applied. Some benchmarks (ADI, NW, LUD, and JCB1D) show particularly high performance when the L1 data cache is not used. Unlike the policy that completely excludes L1 data cache access, the proposed technique should allow cache access during the sampling period. Therefore, performance is not as good as the policy which completely bypasses L1 data cache. LUD benchmark shows 72% miss rate on average, however almost all requests are bypassed in the L1 data cache even if the threshold k is set to 0.9. The reason for this is the case where the miss rates for each kernel are different or the cache is rarely accessed due to workload characteristics. In this experiment, our technique bypasses most of requests when few cache accesses occur for sampling period. From these results, we can know that the miss rate-based bypassing has limitations. It is difficult to find optimal threshold value for all benchmarks. Moreover, it is unreliable to determine the bypassing based on the sampled miss rate. As a result of applying fixed threshold values to all benchmarks, the GPU performance can be improved by 2.5% on average compared to the GPU without bypassing when we use threshold k with 0.9.

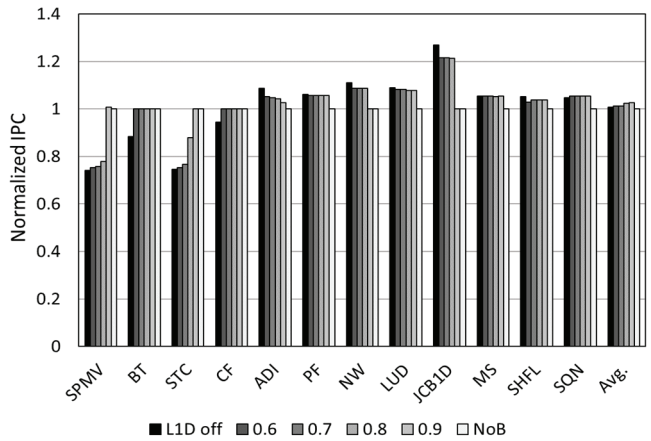

Fig. 3. Performance of miss rate-based bypassing with different thresholds.

Fig. 4 presents the GPU performance of our two-level bypassing technique compared with MRPB bypassing [1], APCM [11], PEMP [8], and CARB [3] techniques. The IPC results are normalized to the GPU architecture without bypassing, which turns on L1 cache always. For SPMV, BT, STC, and CF, the performance of the GPU is degraded when the L1 data cache is turned off. SPMV, BT, and STC represent ambiguous miss rate range which is defined by our study. However, these three benchmarks show at least 84% warp occupancy, i.e., there are many warps that can be executed concurrently to hide long memory latency. In this case, our proposed technique encourages cache access based on the decision considering high TLP when it is difficult to predict the cache gains depending on miss rate. Therefore, the proposed method shows similar performance compared to the no-bypassing scheme whereas previous bypassing techniques presents performance gap. Our proposed technique determines cache bypassing based on miss rate only for the benchmarks which show high miss rate such as PF (99%) and MS (95%). Therefore, the proposed technique improves the performance by 4.6% and 5.6% for PF and MS compared to the no-bypassing scheme, respectively. The other benchmarks (ADI, NW, LUD, JCB1D, SHFL, and SQN) that favorable for bypassing show warp occupancy less than 46%. Although these benchmarks present

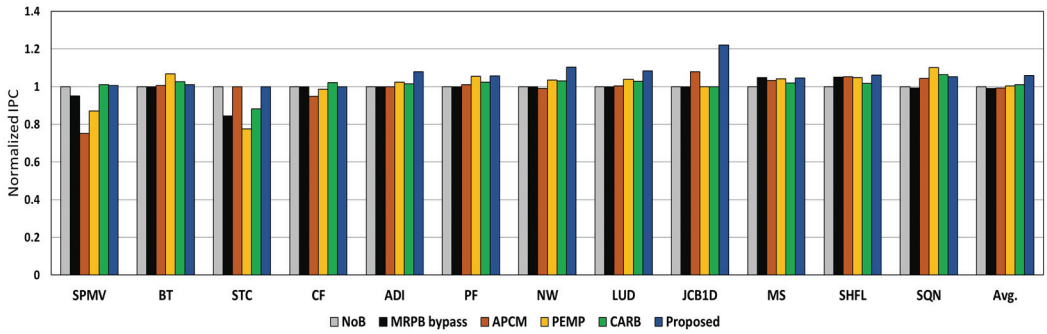

Fig. 4. Performance comparison of bypassing techniques.

ambiguous miss rate range, our technique can predict whether the workload benefits from the cache or not, by considering TLP information. Therefore, the dynamic bypassing decision operates correctly according to the characteristics of the workload. JCB1D benchmark shows significant performance improvement by bypassing L1 data cache. Bypassing on JCB1D benchmark is quite beneficial to GPU performance because the TLP is considerably low and the cache miss rate is relatively high compared to other benchmarks. As a result, our dynamic bypassing technique based on two-level decision achieves 6.1% IPC improvement on average compared to the no-bypassing scheme. According to our observation, if the sampling period is relatively long as compared with the period in which the bypass policy is applied, the performance is degraded even as our technique determines bypassing correctly at all time. This is because the cached data for the sampling period is not re-referenced in the B mode. As a result, setting the sampling period as short as possible helps to improve the GPU performance when L1 data cache is inefficient.

Fig. 5 shows the number of L1 data cache accesses and misses reduced by the proposed technique. Each bar is normalized to the number of cache accesses of the architecture without bypassing. As shown in the graph, reduced rate in the number of accesses and misses are similar to each other. The proposed technique for NW and LUD increases the miss rate by 15% and 36% compared to the no-bypassing scheme. However, since the number of L1 data cache accesses is reduced by 95% in the proposed technique, the impact of increased cache miss rate on the GPU performance can be ignored. Since CF shows very low miss rate (4.6%), our technique simply maintains caching mode based on the first step of proposed technique. As shown in this graph, on average, 60% of requests are bypassed for L1 data cache across the benchmarks. For SPMV, BT, STC, and CF benchmarks, nearly all requests are cached by our proposed technique. According to our observation, if the sampling period is too long as compared with the period in which the bypass policy is applied, the performance is degraded even as our technique determines bypassing correctly at all time. This is because the cached data for the sampling period is not re-referenced in the B mode. As a result, setting the sampling period as short as possible helps to improve the GPU performance when L1 data cache is inefficient. The results for the benchmarks (ADI, PF, NW, LUD, JCB1D, MS, SHFL, and SQN) which prefer B mode show that the percentage of the requests bypassed is 90% on average. As a result, for these benchmarks, all memory requests are bypassed at L1 data cache except for sampling period. The proposed scheme increases the cache miss rate by 6.9% on average. The miss rate increases for nearly all benchmarks even when the performance is improved. Because our technique does not aim to improve cache efficiency itself, cache

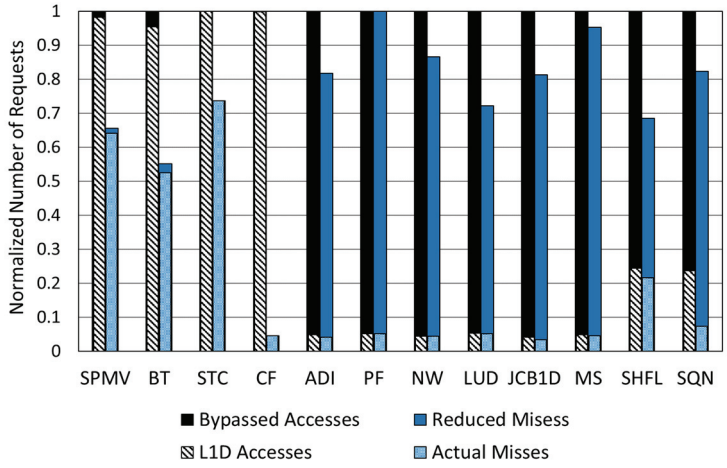

Fig. 5. Access and Misses reduced by the proposed technique.

utilization can be the same or degraded, as our technique bypasses for a specific period when the running workload is not suitable for caching. LUD particularly shows high increase in miss rate of 36%. The miss rate of LUD belongs to the ambiguous miss rate range, which is defined in this paper, however, most memory requests are bypassed in the L1 data cache. Since the average warp occupancy of LUD benchmark is 18% ($w \fallingdotseq 5.8$), the proposed technique determines that the L1 data cache is inefficient. For several benchmarks in our experiment, cache miss rate increases slightly by our technique, however it does not affect performance since the number of total cache accesses is considerably reduced.

5. Conclusion

In this study, we analyzed the cache benefits to GPU performance across different kinds of workloads. Our study revealed that predicting the usefulness of the L1 data cache on GPU performance based only on cache miss rate is difficult. When the captured cache locality is neither obviously high nor low, the performance gain from the L1 data cache must be determined by other factors. To resolve this problem, we proposed a new two-level bypassing method that determines cache access based on two metrics. Unlike previous GPU architectures, the latest GPU cache is beneficial for high TLP workloads. Therefore, we compensate the drawbacks of the miss rate metric by measuring the cache throughput. Our experimental results showed that the proposed technique outperforms the GPU architecture without bypassing by 6% since our technique can effectively and selectively bypass the L1 data cache when the cache is harmful to GPU performance.

Acknowledgement

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2018R1A2B6005740).

References

[1] W. Jia, K. A. Shaw, and M. Martonosi, "MRPB: memory request prioritization for massively parallel processors," in *Proceedings of 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Orlando, FL, 2014, pp. 272-283.

[2] NVIDIA Corporation, "NVIDIA Tesla P100: GP100 Pascal Architecture," 2016 [Online]. Available: https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf.

[3] C. T. Do, J. M. Kim, and C. H. Kim, "Application characteristics-aware sporadic cache bypassing for high performance GPGPUs," *Journal of Parallel and Distributed Computing*, vol. 122, pp. 238-250, 2018.

[4] J. Zhang, Y. He, F. Shen, and H. Tan, "Memory-aware TLP throttling and cache bypassing for GPUs," *Cluster Computing*, vol. 22, no. 1, pp. 871-883, 2019.

[5] NVIDIA Corporation, "NVIDA Tesla V100 GPU architecture," 2017 [Online]. Available: http://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.

[6] M. Gebhart, S. W. Keckler, B. Khailany, R. Krashinsky, and W. J. Dally, "Unifying primary cache, scratch, and register file memories in a throughput processor," in *Proceedings of 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, Vancouver, Canada, 2012, pp. 96-106.

[7] X. Xie, Y. Liang, Y. Wang, G. Sun, and T. Wang, "Coordinated static and dynamic cache bypassing for GPUs," in *Proceedings of 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Burlingame, CA, 2015, pp. 76-88.

[8] C. T. Do, J. M. Kim, and C. H. Kim, "Early miss prediction based periodic cache bypassing for high performance GPUs," *Microprocessors and Microsystems*, vol. 55, pp. 44-54, 2017.

[9] X. Chen, L. W. Chang, C. I. Rodrigues, J. Lv, Z. Wang, and W. M. Hwu, "Adaptive cache management for energy-efficient GPU computing," in *Proceedings of 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, Cambridge, UK, 2014, pp. 343-355.

[10] A. Sethia, D. A. Jamshidi, and S. Mahlke, "Mascar: speeding up GPU warps by reducing memory pitstops," in *Proceedings of 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Burlingame, CA, 2015, pp. 174-185.

[11] G. Koo, Y. Oh, W. W. Ro, and M. Annavaram, "Access pattern-aware cache management for improving data utilization in GPU," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, Toronto, Canada, 2017, pp. 307-319.

[12] J. Fang, X. Zhang, S. Liu, and Z. Chang, "Miss-aware LLC buffer management strategy based on heterogeneous multi-core," *The Journal of Supercomputing*, vol. 75, no. 8, pp. 4519-4528, 2019.

[13] M. Khairy, A. Jain, T. M. Aamodt, and T. G. Rogers, "A detailed model for contemporary GPU memory systems," in *Proceedings of 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Madison, WI, 2019, pp. 141-142.

[14] NVIDIA Corporation, "NVIDIA GeForce GTX 1080," 2016 [Online]. Available: https://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_1080_Whitepaper_FINAL.pdf.

[15] Z. Jia, M. Maggioni, B. Staiger, and D. P. Scarpazza, "Dissecting the NVIDIA Volta GPU architecture via microbenchmarking," 2018 [Online]. Available: https://arxiv.org/abs/1804.06826.

[16] M. Bari, L. Stoltzfus, P. Lin, C. Liao, M. Emani, and B. Chapman, "Is data placement optimization still relevant on newer GPUs?," 2018 [Online]. Available: https://www.osti.gov/servlets/purl/1489476.

[17] A. Karki, C. P. Keshava, S. M. Shivakumar, J. Skow, G. M. Hegde, and H. Jeon, "Tango: a deep neural network benchmark suite for various accelerators," in *Proceedings of 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Madison, WI, 2019, pp. 137-138.

Gwang Bok Kim **https://orcid.org/0000-0002-6750-1008**

He received the B.S. and M.S. degrees in electronics and computer engineering from Chonnam National University, Gwangju, Korea in 2013 and 2015, respectively. He received the Ph.D. degree in computer engineering from Chonnam National University in 2019. Now he is working at R&D Center 2, SFA Engineering, Korea. His research interests include computer architecture, low power systems, and GPGPU.

Cheol Hong Kim **https://orcid.org/0000-0003-1837-6631**

He received the B.S. degree in Computer Engineering from Seoul National University, Seoul, Korea in 1998 and M.S. degree in 2000. He received the Ph.D. in Electrical and Computer Engineering from Seoul National University in 2006. He worked as a senior engineer for SoC Laboratory in Samsung Electronics, Korea from December 2005 to January 2007. He worked as a Professor at Chonnam National University, Korea from 2007 to 2020. Now he is working as a Professor at School of Computer Science and Engineering, Soongsil University, Korea. His research interests include computer systems, embedded systems, computer architecture, and GPGPU.