Regular Paper

Triple-helix Writeback: Reducing Inner-Bankgroup Contention in DDR4 SDRAM Memory System

Kohei Hosokawa[1,a)] Yasuo Ishii[1,b)] Kei Hiraki[1,c)]

Abstract: For decades, memory controllers for DDRx memory exploit row-buffer locality to improve the efficiency of memory access. Recently, the bankgroup feature of the latest DDR4 memory requires inter-bankgroup parallelism to saturate the memory bus unlike previous DDRx generations. To exploit both the row-buffer locality and the inter-bankgroup parallelism, the DDR4 memories require intelligent memory controllers.
Modern memory controllers reorder both read requests and write requests to maximize the available memory bandwidth. Compared with read requests, write requests have large scheduling space because write requests do not stall the processor pipeline unlike read requests. Therefore, several memory scheduling techniques utilize the dirty data on the last-level caches to exploit the row-buffer locality. However, these techniques cannot utilize the inter-bankgroup parallelism effectively. To maximize the memory access efficiency on the DDR4 memory access, we propose Triple-helix Writeback. Triple helix writeback analyzes the writeback request for the last-level cache to detect multiple writeback streams to utilize the inter-bankgroup parallelism. When multiple writeback streams for different bankgroups are detected, the memory controller eagerly writes the dirty data on the detected writeback streams simultaneously. This scheduled writeback method improves the memory bus utilization efficiency of complicated DDR4 memories because the memory controller can effectively exploit both the row-buffer locality and the inter-bankgroup parallelism.
We evaluated Triple-helix Writeback by the cycle-accurate full system simulation. The simulation result shows that the Triple-helix Writeback improves performance and energy efficiency by 4.7% and 5.9%, respectively.

Keywords: cache, DRAM, memory scheduling, scheduled writeback, power efficiency

1. Introduction

As the semiconductor process technologies have been improved, the performance gap between processor chips and memory system has increased. Recently, the performance of the memory system becomes one of the major performance bottlenecks of the computer systems [10]. Moreover, the energy consumption of the memory system is also the major constraint of the modern processor design. For example, the energy consumption of the memory system accounts for 20% or more of typical computer systems [11]. To maximize the performance and to minimize the energy consumption of the memory system, modern processor chips employ intelligent memory controllers which realize novel memory access scheduling.

Modern memory access scheduling features are designed to resolve following two difficulties which are (a) *3-dimentional hierarchical structure* and (b) *write-induced penalty*. As for (a) the *3-dimentional structure* [12], the DDRx memory contains independent memory banks. Each bank has the row-buffer which is the data buffer to store the recently accessed data. This structure requires spatial and temporal locality of memory requests because the controller has to switch the data on the row-buffer to access the data on the other row. This situation reduces the memory performance and increases the energy consumption. To avoid such situations, the memory controller searches the memory requests whose target data is already stored on the row-buffer. To exploit such memory requests, the memory controller employs requests buffer. Typically, the size of the buffer is limited (approx. 32 entries) because of the implementation cost and the design complexity. On such small scheduling buffer, the efficiency of the memory access scheduling is also limited. As well as the complicated data structure, (b) the *write-induced penalty* has a large effect on the system performance. Write induced penalty is a bus turnaround penalty between a read operation and a write operation. As is shown in the datasheet of DDR3 [1] and DDR4 [2], the write request for the memory system causes large penalty cycles when the following request is a read request. To minimize this write induced penalty, the memory controller should issue several successive write requests at a time to reduce the bus switching frequency. The low switching frequency improves the available memory bandwidth of the memory intensive workloads. Modern memory controller has to consider these complicated timing constraints.

To resolve these two difficulties, DRAM-aware writeback techniques have been proposed [7]. DRAM-aware writeback is scheduled writeback method [8] which searches the dirty cache lines on the last-level cache (LLC). When the useless dirty cache lines for one row-buffer are detected, the detected data is written back to the main memory. This DRAM-aware writeback

[1] The Universit of Tokyo, Hongo, Tokyo 113–0033, Japan
[a)] khosokawa@is.s.u-tokyo.ac.jp
[b)] yishii@is.s.u-tokyo.ac.jp
[c)] hiraki@is.s.u-tokyo.ac.jp

Latency	Symbol	DRAM cycles	Latency	Symbol	DRAM cycles	Latency	Symbol	DRAM cycles
Read / Write	tBL	4	Additive Latency	AL	0	Activate to Activate	tRC	39
Precharge	tRP	11	Activate to Precharge	tRAS	28	Read to Precharge	tRTP	6
Activate	tRCD	11	Burst Length	tBL	4	Write recovery	tWR	12
			Row to Row delay	tRRD	6	Column strobe to column strobe	tCCD	4
			Four activate windows	tFAW	24	Write to read	tWTR	6

Table 1 DDR3 1600 SDRAM timing specification

increases the row-buffer locality and reduces the write induced penalty because the writeback sequence can be searched from many dirty cache lines while the conventional methods have to select write requests from the requests on the write buffer. This method works well on the DDR3 SDRAM memory system, but does not work appropriately on DDR4 SDRAM system because of the bankgroup feature. On DDR4 SDRAM, four memory banks make a bankgroup. There are 2 or 4 bankgroups in one DDR4 memory device. When the memory controller accesses the data on one bankgroup, the controller waits between consecutive two memory requests. This penalty is called inner-bankgroup penalty. Due to this penalty, the bus utilization ratio cannot exceed the 66% of the peak memory bandwidth when the memory controller accesses only one bankgroup.

To avoid inner-bankgroup penalty, two or more bankgroups should be accessed simultaneously to increase the available memory bandwidth efficiently. For this restriction, memory controller should exploit not only row-buffer locality, but also inter-bankgroup parallelism. DRAM-aware writeback does not consider inter-bankgroup parallelism due to its locality-oriented structure. Therefore, the available memory bandwidth is often significantly reduced for the DDR4 memory devices.

To consider both of row-buffer locality and inter-bankgroup parallelism, we propose *Triple-helix Writeback*. Triple-helix Writeback analyzes the dirty cache lines on the LLC and detects the writeback sequences. Based on this analysis result, Triple-helix Writeback issues two or more writeback sequences intertwined with each other. Triple-helix Writeback exploits the row-buffer locality and the inter-bankgroup parallelism simultaneously. Our scheduled writeback technique can increase the available memory bandwidth even if the number of write-buffer on memory controller is small. Our evaluation shows that Triple-helix Writeback improves system performance and energy consumption by 4.7% and 5.9% compared to the baseline, respectively. Especially for memory intensive workloads, our method improves the performance and the energy efficiency of the DDR4 memory systems.

This paper is organized as follows. In Section 2, background and our motivation of this paper is presented. In Section 3, we propose the Triple-helix Writeback. In Section 4, we evaluate the Triple-helix Writeback technique. Finally, we conclude this paper in Section 5.

2. Background and Related Work

2.1 DRAM Memory System

Modern DRAM devices realize high clock frequency for its memory bus. However, it also requires various complicated timing constraints to access DRAM data cells. Moreover, next generation DDR device, which is called DDR4 SDRAM, employs more complicated timing constraints to increase the memory bandwidth much more.

Table 1 shows the representative timing constraint of the latest DDR3 device. The DDR3 devices can process one read / write request for every four DRAM clock cycles (*tBL*). On the other hand, a precharge and activation takes much longer times (*tRP* and *tRCD*). When the read request accesses the data already on the row-buffer, the memory requests can eliminate the activate requests and reduces the memory access latency. Moreover, this row-hit memory access also reduces the energy consumption for activation and precharge operations. Therefore, the row-buffer locality of the DRAM devices is essential for the memory access scheduling to improve the performance and the energy efficiency.

As well as the timing constraints corresponding to the row-buffer locality, the other timing constraints also affect the sustained performance of main memory. The bus turnaround penalty of read and write operation is one of such constraints. When a read request is issued just after write requests, the read request waits for *tWTR* cycles. This penalty is longer than that of *tBL*. This long penalty is required to guarantee that the write requests do not conflict with the read request.

Moreover, DDR4 SDRAM memory system adds bankgroup feature to increase the memory bandwidth. The bankgroup is the structure that contains four memory banks. A bankgroup needs two blank cycles between two successive read/write operations for the same bankgroup, so it can serve up to 66% of the memory data bus bandwidth throughput. To saturate the memory data bus efficiently, the memory controller must issue read/write operations to two or more bankgroups simultaneously. Therefore, the memory accesses with excessively high locality and poor parallelism cannot be processed effectively by the DDR4 SDRAM memory system. As is shown, the bankgroup feature adds the new timing constraints in addition to the constraints of DDR3 SDRAM. We call this new timing constraints *inner-bankgroup penalty*. Figure 2 shows the inner-bankgroup penalty of the DDR4 memory devices. As shown in the figure, the following memory requests suffers inner-bankgroup penalty. To avoid this penalty, the memory controller has to issue two memory request streams which access different bankgroups.

2.2 Memory access scheduling

To increase the available memory bandwidth of complicated DDRx SDRAMs, Modern processor chips employ intelligent memory access schedulers. The memory access schedulers control row-buffer locality by reordering memory requests on the scheduling buffer. This component increases the available memory bandwidth and reduces the energy consumption.

On memory access scheduling, the asymmetric behavior between the read and write operation is important. The read operation causes the processor core to stall immediately, so they limit the system performance. On the other hand, the write operation does not directly stall the core pipeline, so it can be postponed for a while. However, the write operations heavily interfere with the read operations due to the write-induced penalty. To alleviate the bandwidth consumption and interference of write requests, the write requests are typically suspended in the write queue, and flushed at a time when the write queue spills [7]. The size of the write queue is limited due to the full-associative structure, so it does not have a large scheduling space and causes the frequent spill on the memory-intensive workloads. For the more scheduling space on the write queue, the Virtual Write Queue [9] and the Decoupled Last Write Prediction [13] was proposed.

2.3 Scheduled Writeback

Stuecheli et al. [9] proposed the concept of the *scheduled writeback*. This concept means that not the memory controller but the LLC controller should schedule when to issue the writeback of the dirty blocks. Without the writeback scheduling, the dirty cache lines are written back on eviction (mainly due to the cache miss by a read request), which causes a read and write memory accesses contending with each other. The scheduled writeback issues the writeback requests in advance of the eviction to avoid the contention and exploit the locality.

The DRAM-aware writeback [7] is one of such scheduled writeback methodology. The DRAM-aware writeback exploits the spatial and temporal locality of the write accesses by issuing the writeback sequences eagerly from the LLC. Moreover, the DRAM-aware writeback enhances the granularity of the write access sequence, so it can reduce the overhead from the row-buffer switching between a write and a read operation. Therefore, the DRAM-aware writeback can utilize the data bus bandwidth efficiently.

On the DDR4 SDRAM environment, however, the DRAM-aware writeback method suffers from a significant performance loss due to the inner-bankgroup penalty. The DRAM-aware writeback method has highly row-buffer locality, but the bankgroup parallelism is not considered. For this reason, only one bankgroup serves the write requests on the progress of the scheduled writeback operation, so up to 33% of the memory data bus bandwidth is wasted by the heavy inner-bankgroup penalty. Therefore, even if the row-buffer hit rate keeps high, the total bus bandwidth usage rate gets low. The inter-bankgroup round-robin approach does not work appropriately on this problem (Figure 1(a)). The round-robin algorithm does not reduce the unevenness of the dirty cache lines, so it cannot avoid the situation in which only one bankgroup is working.

3. Triple-helix Writeback

3.1 Design Overview

We propose Triple-helix Writeback. This is scheduled writeback method to recognize the LLC dirty blocks and writeback them. Compared to the existing scheduled writeback methods, Triple-helix Writeback consider not only row-buffer locality but also inter-bankgroup parallelism. For this reason, Triple-helix Writeback can work well on DDR4 SDRAM.

Triple-helix Writeback monitors write access to LLC and recognizes their spatial locality. Triple-helix Writeback realizes this locality recognition by analysis of write accesses (new-dirty-entry event) and dirty cache line eviction (evict-dirty-entry event) on LLC. After the specification of the locality, Triple-helix Writeback issues scheduled writeback eagerly to clean dirty cache lines. On the writeback, Triple-helix Writeback schedules writebacks so that (1) multiple bankgroups are always active (2) row-buffer locality is kept high in each bankgroup. With this strategy, Triple-helix Writeback enhances both inter-bankgroup parallelism and row-buffer locality.

3.2 Components and Structure

Triple-helix Writeback employs two components: (1) *WST table* which analyses row-buffer locality of LLC write access for each bankgroup, (2) *Writeback Arbiter* which schedules LLC looking up and writeback operation. We will explain these components below.

(1) WST table is row-buffer locality tracking table. There is one corresponding WST table per one memory channel. A WST table consists of multiple BT tables. One BT table corresponds to one bankgroup. BT table monitors LLC regions which receives many write accesses. A BT entry is the statistics entry of one monitoring region. The size of this monitoring region is equal to a row address. BT entry has 3 data fields: (a) Target Row Address (b) Confidence counter (c) dirty-entry counter. (a) is 64 bit field which stores the monitoring target address. (b) is 4 bit saturating counter for the replacement algorithm of BT entry. This is used to allocate the hotter spot. (c) is 8 bit saturating counter to store the actual number of dirty cache lines in the monitoring region.

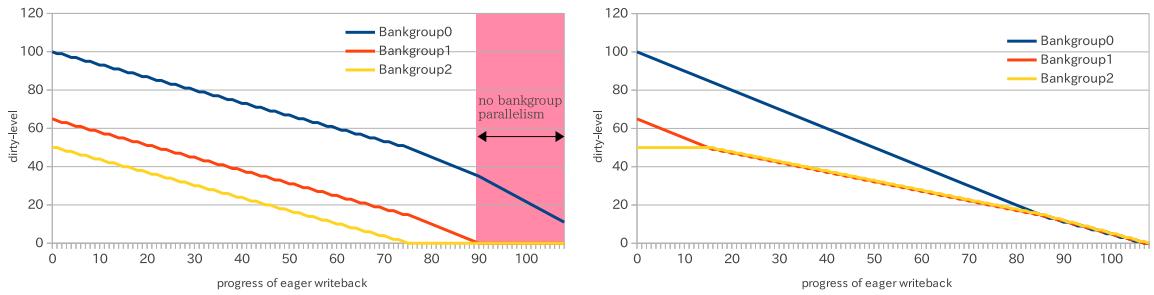
(2) Writeback Arbiter is scheduler structure to control writeback operation. Writeback Arbiter starts scheduled writeback process when LLC gets more dirty cache lines than a specified watermark. At the beginning of scheduled writeback, Triple-helix Writeback estimates the number of dirty cache lines of each bankgroup respectively. Then, two bankgroups search dirty cache lines in LLC in parallel and issue scheduled writeback. As a result, Triple-helix Writeback can exploit the inter-bankgroup parallelism and row-buffer locality.

3.3 Writeback algorithm

Triple-helix Writeback execution consists of two parts: (p) locality analysis part and (q) scheduled writeback part.

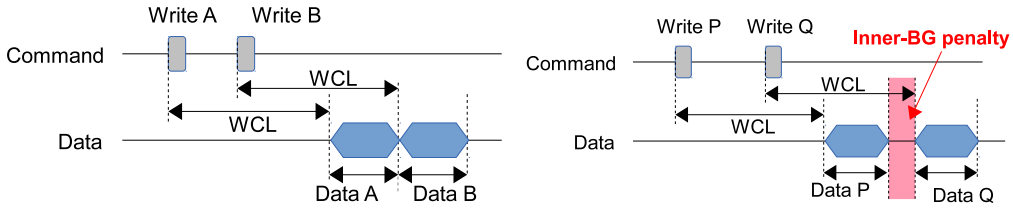
(p) In locality analysis part, WST table updates BT entries by monitoring the state transition of dirty cache lines in LLC. If new-dirty-entry (or evict-dirty-entry) event happens in the monitoring address region of a BT entry, the confidence counter and dirty-entry counter of the corresponding BT entry is incremented (or decremented) by 1. If new-dirty-entry happens outside of all the monitoring address regions, then

(a) **when the target bankgroup has one or more BT entry with their confidence counter 0**: Choose one of these BT entries and overwrite it. Target Row address of the new BT entry is the new dirty cache line address, confidence counter is 1, and dirty-entry



(a) Round-robin (b) Triple-helix

Fig. 1 Transition of the number of dirty cache lines in each bankgroup. The round-robin algorithm does not aware the unevenness of the dirty-lebel of bankgroups among bankgroups. As a result, it suffers from inner-bankgroup penalty. On the other hand, Triple-helix Writeback schedules the writeback operation to even-out the number of dirty cache lines of bankgroups.



(a) Writes on DDR3 SDRAM (b) Writes on DDR4 SDRAM

Fig. 2 Sequential write access to the same row-buffer. In DDR3 SDRAM configuration, the successive read/write access to the same row-buffer does not cause timing restriction. However, on DDR4 SDRAM system, the successive access to the same bankgroup causes extra penalty cycles.

counter is 1.

(b) **when all the BT entries of the target bankgroup have their confidence counter more than 0**: Decrement all the confidence counters of the target bankgroup by 1. To reserve spatial locality, confidence counter is saturated if dirty-entry counter is more than 16.

(q) In scheduled writeback part, Writeback Arbiter scans LLC based on WST table and issues scheduled writeback. The operation flow is shown in Figure 3. This operation is executed independently of each memory channel.

(1): Start scheduled writeback operation on the condition as follows: (a) write-buffer of memory controller gets saturated (b) any memory rank starts refreshing (c) the number of BT entries with (dirty-entry counter) $\geq$ 16 gets more than a watermark (experimentally, we set this watermark as 1/4 of total BT entries).

(2): Estimate the number of dirty cache lines of each bankgroup by summing up dirty-entry counters.

(3): look up a BT entry of the bankgroup with the most and the second most dirty cache lines respectively. The Target Row Addresses of these two BT entries are used as the search target regions.

(4)/(5): Find one LLC dirty cache line from each of the search target region respectively. If no dirty cache line is found in either of search target region, delete the corresponding BT entry and recalculate the search target region.

(6): If a corresponding dirty cache line to the two search target regions is found respectively, writebacks both of them to memory controller. After that, decrements the corresponding dirty-entry counter by 1. Then return to step (2).

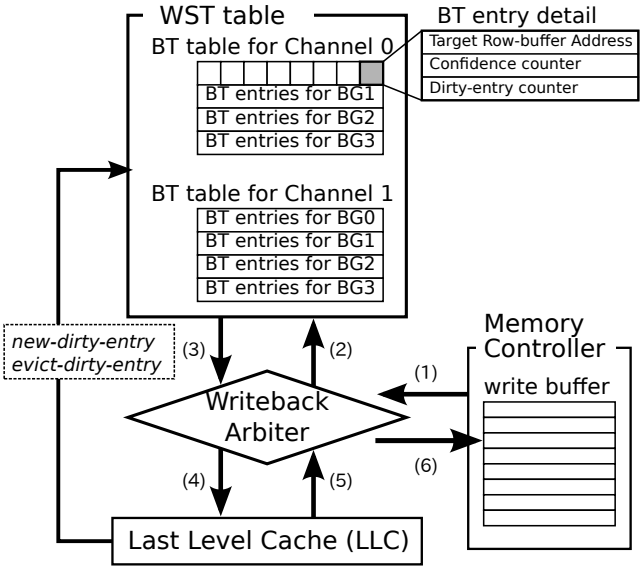


Fig. 3 Structure of the Triple-helix Writeback.

By this algorithm, the bankgroup with the most dirty cache lines is served scheduled writeback operation preferentially as is shown in Figure 3. As a result, different from the round-robin algorithm, Triple-helix Writeback can reduce bias between bankgroups as the progress of writeback operation. Therefore, Triple-helix Writeback completes scheduled writeback with keeping inter-bankgroup parallelism.

4. Evaluation

We evaluated the Triple-helix Writeback on 16-core proces-

Execution Core	4.8GHz, 16core CMP, out of order, 256 entry reorder buffer, 48 entry load queue 44 entry store queue, 4 width issue/decode. 15 stages 256 physical registers
Caches	L1 I-cache: 64KB, 8way, private, 64 bytes block size, 2-cycle, LRU L1 D-cache: 64KB, 8way, private, 64 bytes block size, 2-cycle, LRU L2 cache: 8MB/8way, shared, 64 bytes block size, 14-cycle, LRU
DRAM	2 memory channels and controllers, 2 ranks per channel, burst length 8, open page policy, FR-FCFS 16 banks (=4 bankgroups) per channel, 8K bytes row-buffer per bank, DDR4-2400[2]

Table 2 System Configuration

Latency	Symbol	DRAM cycles	Latency	Symbol	DRAM cycles
Additive Latency	AL	0	Activate to Activate	tRC	57
Activate to Precharge	tRAS	39	Read to Precharge	tRTP	9
Burst Length	tBL	4	Write recovery	tWR	18
Activate to Activate (different bank)	tRRD	l: 6 s: 4	Column address strobe to column address strobe	tCCD	l: 6 s: 4
Four activate windows	tFAW	26	Write to read	tWTR	l: 9 s: 3

Table 3 DDR4 2400 SDRAM timing specification.

The l or s value in the *DRAM cycles* columns means the inner- and inter-bankgroup specification. When two successive operations are issued to the same bankgroup, the longer (l value in this table) cycles are needed.

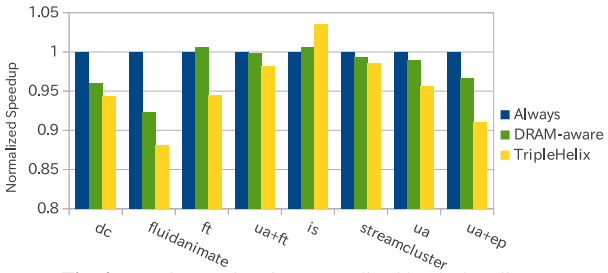


Fig. 4 Total execution time normalized by the baseline.

sor environment by using MARSSx86 cycle-accurate full-system simulator [4] with DRAMSim2 memory simulator [5]. We modified the DRAMSim2 simulator to evaluate DDR4 SDRAM system based on the DDR4 SDRAM standard datasheet of Micron [1]. The memory address mapping scheme is column centric [12]. Table 2 shows the simulation parameters on this evaluation. We employ small 32-entry read/write buffers on memory controller [6].

We used the memory-intensive workloads of NAS parallel benchmark programs [3] and PARSEC benchmark programs [14]. We executed these workloads on 16-threads configuration as is shown in table 2. We fast-forwarded all the benchmarks to skip the initialization phase, and evaluated until all processor cores finish the following 100M instructions. The system performance was calculated based on the total execution cycles on the cycle-accurate simulator. The total energy consumption was estimated based on the DDR4 SDRAM standard datasheet of Micron [2]. We compared the Triple-helix Writeback with two other memory and LLC control scheme: the *baseline* which does not use the eager writeback, and the *DRAM-aware* which utilizes the existing DRAM-aware writeback. The configuration of the evaluation environment is shown in table 2. We used streaming prefetch method on each cache level, and employed the LRU replacement policy as the cache line management policy.

4.1 Simulation Result

Figure 4 shows the total execution time normalized by the baseline performance. As shown in the figure, Triple-helix Writeback outperforms the configurations expect for IS benchmark. Compared with baseline configuration, Triple-helix Writeback improves the performance by 12.0% for fluidanimate. On average,

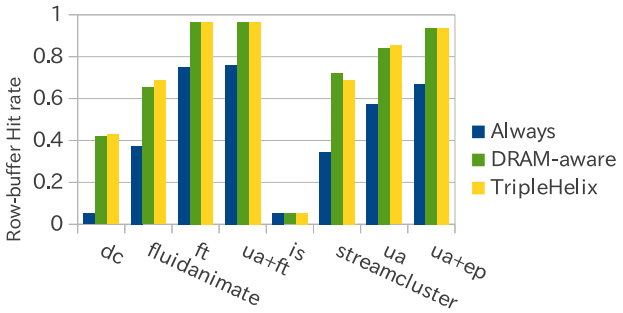


Fig. 5 Row-hit ratio for write requests.

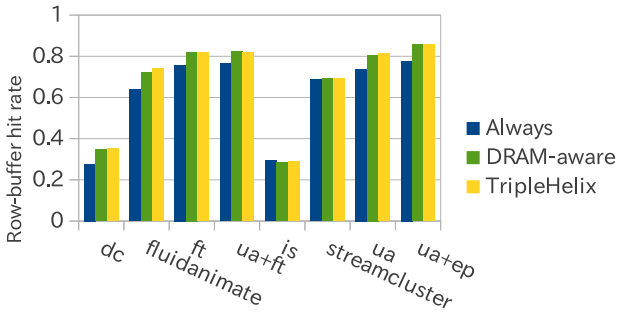


Fig. 6 Row-hit ratio for all memory requests.

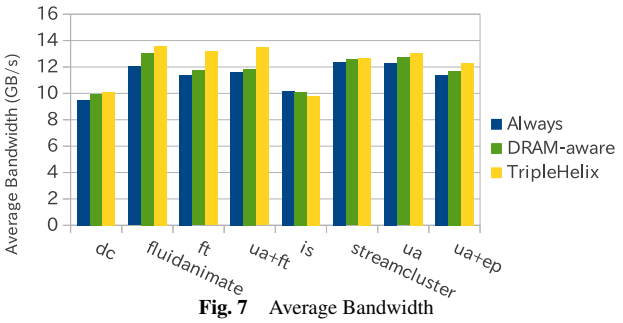


Fig. 7 Average Bandwidth

Triple-helix Writeback reduces the execution time by 4.7% from baseline configuration. To analyze the performance contributions in detail, we evaluate the other performance factors.

Figure 5 and Figure 6 shows the row-buffer hit rate. High row-hit ratio reduces the energy consumption for activate / precharge and improve the available memory bandwidth. Compared with baseline configuration, Triple-helix Writeback increases the row-hit ratio for write requests by 25.1%. For all memory requests,

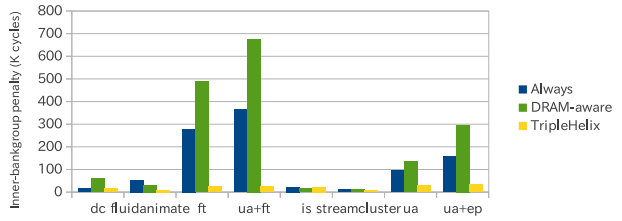


Fig. 8 Total Inner-Bankgroup penalty

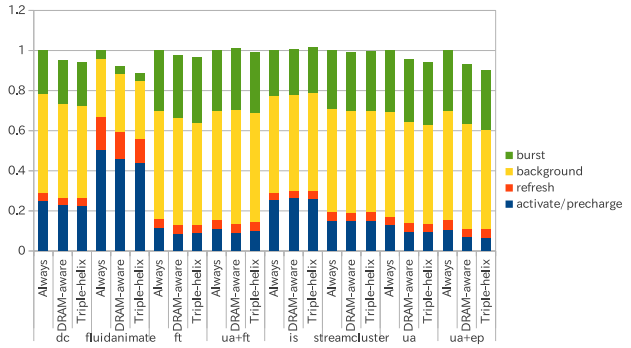


Fig. 9 Normalized Total Energy Consumption

Triple-helix Writeback increases the row-hit ratio by 5.6%. As shown in the figures, the Triple-helix Writeback cannot increase the row-hit for IS benchmark. This is the main reason why Triple-helix Writeback cannot improve the performance for IS benchmark.

Figure 7 is the average memory bandwidth. Triple-helix Writeback consistently outperforms the other configurations expect for IS benchmark. On average, Triple-helix Writeback increase the available memory bandwidth by 8.2%. Reduction of inner-bankgroup penalty contributes the performance improvement.

To evaluate the inter-bankgroup parallelism, we count the bus idle time due to the inner-bankgroup penalty. Figure 8 shows the evaluation result of counted penalty cycles. As shown in the figure, Triple-helix Writeback reduces the penalty cycles. For UA+FT, the inner-bankgroup penalty is reduced by 93.1% from baseline configuration.

Figure 9 shows the total energy consumption normalized by baseline configuration. Triple-helix Writeback reduces the energy consumption by 11.6% for fluidanimate. On average, Triple-helix Writeback reduces the energy consumption by 5.9%. High row-hit ratio reduces the activate energy and high performance reduces the background energy. As shown in this section, Triple-helix increases the performance and reduces the energy consumption compared with existing memory access scheduling techniques. Its performance improvement is supported by both row-buffer locality and inter-bankgroup parallelism.

5. Conclusion

In this paper, we propose Triple-helix Writeback, a novel memory control scheme for the DDR4 SDRAM memory system. Triple-helix Writeback combines the inter-bankgroup parallelism and the row-buffer locality effectively to enhance the throughput and energy efficiency. The Triple-helix Writeback scheme analyses the memory request on the last level cache and recognizes the memory access locality of write requests. By using the analyzing result, Triple-helix Writeback issues the multiple write sequences intertwined with each other to exploit both the row-buffer locality and the bankgroup parallelism. Existing DRAM-aware writeback suffers from the heavy inner-bankgroup penalty which arises on DDR4 SDRAM system. Our Triple-helix Writeback methodology can solve this problem by exploiting the inter-bankgroup parallelism.

Evaluation results show that Triple-helix Writeback can improve the inter-bankgroup parallelism while exploiting the row-buffer level locality. We conclude that the high row-hit rate and the high DRAM memory bus utilization ratio improve not only the total execution time but also the energy efficiency. As a future work, we will unify the inter-bankgroup parallelism and row-buffer locality not only the writeback operations but also the read operations for the further performance improvement.

References

[1] JEDEC. JEDEC Standard: DDR3 SDRAM STANDARD (JESD79-3D). http://www.jedec.org/standards-documents/docs/jesd-79-3d.

[2] JEDEC. JEDEC Standard: DDR4 SDRAM STANDARD (JESD79-4). http://www.jedec.org/standards-documents/results/jesd79-4 ddr4.

[3] D. H. Bailey et al. *NAS parallel benchmarks.* Technical report, NASA Ames Research Center, 1994.

[4] A. Patel, F. Afram, S. Chen, and K. Ghose. *MARSSx86: A full system simulator for x86 CPUs.* In Proceedings of the 2011 Design Automation Conference, June 2011.

[5] P. Rosenfeld, E. Cooper-Balis, and B. Jacob. *Dramsim2: A cycle accurate memory system simulator.* Computer Archi- tecture Letters, PP(99):1, 2011.

[6] Wang, Zhe and Khan, Samira M. and Jiménez, Daniel A. *Improving writeback efficiency with decoupled last-write prediction.* ISCA '12, 2012

[7] C. J. Lee, V. Narasiman, E. Ebrahimi, O. Mutlu, and Y. N. Patt. *Dram-aware last level cache writeback: Reducing write-caused interference in memory system.* In HPS Tech- nical Report, TR-HPS-2010-002.

[8] H.-H. S. Lee, G. S. Tyson, and M. K. Farrens. *Eager writeback - a technique for improving bandwidth utilization.* In Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture, MICRO 33, pages 1121, New York, NY, USA, 2000. ACM.

[9] J. Stuecheli, D. Kaseridis, D. Daly, H. C. Hunter, and L. K. John. *The virtual write queue: coordinating dram and last- level cache policies.* In Proceedings of the 37th annual in- ternational symposium on Computer architecture, ISCA10, pages 7282, New York, NY, USA, 2010. ACM.

[10] D. Kaseridis, J. Stuecheli, and L. K. John. *Minimalist open-page: A dram page-mode scheduling policy for the many-core era.* In MICRO 44, pages 2435, 2011.

[11] Nidhi Aggarwal and Jason F. Cantin and Mikko H. Lipasti and James E. Smith. *Power-Efficient DRAM Speculation.* HPCA 2008. IEEE 14th International Symposium on , vol., no., pp.317,328, 16-20 Feb. 2008.

[12] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, *Memory access scheduling*, In Proceedings of the International Symposium on Computer Architecture, 2000.

[13] Wang, Zhe and Khan, Samira M. and Jiménez, Daniel A. , *Improving writeback efficiency with decoupled last-write prediction*, In Proceedings of the 39th Annual International Symposium on Computer Architecture, 2012

[14] C. Benia et al. *The PARSEC Benchmark Suite: Characterization and Architectural Implications.* Technical report, Princeton University, 2008.