PAPER

Evaluating Energy-Efficiency of DRAM Channel Interleaving Schemes for Multithreaded Programs

Satoshi IMAMURA[†a)], **Yuichiro YASUI**[††], ***Nonmembers***, **Koji INOUE**[†††], **Takatsugu ONO**[†††], ***Members***, **Hiroshi SASAKI**[††††], ***and*** **Katsuki FUJISAWA**[††], ***Nonmembers***

SUMMARY The power consumption of server platforms has been increasing as the amount of hardware resources equipped on them is increased. Especially, the capacity of DRAM continues to grow, and it is not rare that DRAM consumes higher power than processors on modern servers. Therefore, a reduction in the DRAM energy consumption is a critical challenge to reduce the system-level energy consumption. Although it is well known that improving *row buffer locality* (RBL) and *bank-level parallelism* (BLP) is effective to reduce the DRAM energy consumption, our preliminary evaluation on a real server demonstrates that RBL is generally low across 15 multithreaded benchmarks. In this paper, we investigate the memory access patterns of these benchmarks using a simulator and observe that cache line-grained channel interleaving schemes, which are widely applied to modern servers including multiple memory channels, hurt the RBL each of the benchmarks potentially possesses. In order to address this problem, we focus on a row-grained channel interleaving scheme and compare it with three cache line-grained schemes. Our evaluation shows that it reduces the DRAM energy consumption by 16.7%, 12.3%, and 5.5% on average (up to 34.7%, 28.2%, and 12.0%) compared to the other schemes, respectively.

key words: *DRAM, address mapping schemes, energy efficiency*

1. Introduction

As the size of transistors shrinks, the amount of hardware resources equipped on server platforms has been increasing. As a result, state-of-the-art servers can contain hundreds of cores and a few terabytes of DRAM [1]. A larger amount of resources can yield higher performance but also cause higher power consumption. In particular, processors and DRAM typically consume a large fraction of the system-level power. While the power consumption of processors has not been largely increased with decreasing CPU frequency, that of DRAM continues to grow. Since it is not rare that DRAM consumes higher power than processors on modern servers [2], a reduction in the energy consumption of DRAM is a critical challenge to reduce that of an entire system.

In terms of the DRAM energy consumption, there are two important parameters: *row buffer locality* (RBL) [3]–[7] and *bank-level parallelism* (BLP) [8]–[14]. DRAM contains a row buffer, which acts as a cache for the most recently accessed DRAM row, per bank, and memory accesses become faster and more energy-efficient if required data is stored in a row buffer. RBL is generally defined as the spatial data locality in row buffers. On the other hand, current memory systems commonly apply *multi-bank architectures*, which can achieve a high memory bandwidth because multiple banks can be accessed in parallel. This parallelism is called BLP. Although improving these two parameters is effective to reduce the DRAM energy consumption, our preliminary evaluation on a real server demonstrates that RBL is generally low across 15 multithreaded benchmarks.

In this paper, we first investigate the memory access patterns of these benchmarks using a simulator in order to analyze the reason for low RBL. We use a cache line-grained channel interleaving scheme for this investigation, because it is typically applied on modern servers to fully exploit the bandwidth of multiple memory channels [4], [9], [10]. Our analysis reveals that the 15 benchmarks are classified into two groups. The five of them originally expose low RBL due to their irregular memory access patterns. On the other hand, the remaining ten benchmarks potentially possess high RBL, but we obtain the following observations for them. (1) The cache line-grained scheme hurts the RBL of each thread. (2) It efficiently exploits BLP across multiple threads, but not within each thread.

On the basis of these observations, we focus on a row-grained channel interleaving scheme that interleaves contiguous row-sized blocks across memory channels [8], [15]. It can improve RBL of each thread with sustaining high BLP across multiple threads, leading to a performance improvement. Moreover, it can reduce the DRAM power consumption by reducing the number of banks used in parallel. In order to show its effect on the DRAM energy consumption, we quantitatively compare it with three types of cache line-grained schemes.

The contributions of this work are as follows*.

Manuscript received September 19, 2017.
Manuscript revised March 17, 2018.
Manuscript publicized June 8, 2018.

[†]The author is with Fujitsu Laboratories Ltd., Kawasaki-shi, 211–8588 Japan.
[††]The authors are with Institute of Mathematics for Industry, Kyushu University, Fukuoka-shi, 819–0395 Japan.
[†††]The authors are with Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka-shi, 819–0395 Japan.
[††††]The author is with Department of Computer Science, Columbia University, USA.
a) E-mail: s-imamura@jp.fujitsu.com
DOI: 10.1587/transinf.2017EDP7296

*This paper is an extended version of our previous paper [16] that evaluated a row-grained channel interleaving scheme for a graph analysis benchmark on a simulator. In this paper, we evaluate it with various types of multithreaded benchmarks and investigate its impact on a real server.

- Our evaluation on the simulator with the 15 multithreaded benchmarks shows that the row-grained scheme reduces the DRAM energy consumption by 16.7%, 12.3%, and 5.5% on average (up to 34.7%, 28.2%, and 12.0%) compared to three cache line-grained schemes, respectively.
- The sensitivity analysis on the simulator in terms of the numbers of cores and memory channels reveals that the row-grained scheme is effective for various configurations of memory controllers.
- We emulate the row-grained scheme on a real server using a micro-benchmark and demonstrate that its impact depends on the RBL and memory access frequency of a program, but it is effective in various situations.

This paper is organized as follows. Section 2 describes the mechanisms of DRAM and shows the results of our preliminary evaluation. Section 3 introduces related work. Section 4 then analyzes the memory access patterns of the benchmarks and demonstrates the effect of the row-grained scheme. Section 5 and Sect. 6 show evaluation results on the simulator and on the real server, respectively. Finally, we conclude this work in Sect. 7.

2. Preliminary

In this section, we first explain the DRAM mechanisms of modern memory systems and then evaluate the RBL of multithreaded benchmarks on a real server.

2.1 DRAM Mechanisms

Current memory systems typically apply multi-bank architectures, which have conceptual hierarchical structures including channels, ranks, and banks, as illustrated in Fig. 1. A memory controller on a processor manages multiple channels, each of which contains multiple ranks. A rank is a set of multiple banks. Moreover, a bank consists of multiple rows, each of which contains multiple columns corresponding to 64-bit data. The number of banks accessed in parallel (i.e., BLP) is an important parameter for memory bandwidth.

A row buffer is a cache included per bank to store a most recently accessed row in a corresponding bank. There are three types of row buffer accesses, as illustrated in Fig. 2. If a row buffer stores a row containing required data (e.g., row 2 in Fig. 2), it can be accessed immediately. This situation is called a *row buffer hit*. If a row buffer does not store a row, a memory controller must issue an *activate* command to fetch the required row into a row buffer; then, required data can be accessed. This is called a *row buffer miss*. Furthermore, if a row buffer contains a different row from the required one (e.g., row 1 in Fig. 2), the row must be written back to the original row by a *precharge* command; then, the required row is fetched into a row buffer by an activate command. After that, the required data is finally accessed. This situation is called a *row buffer conflict*. In general, RBL is

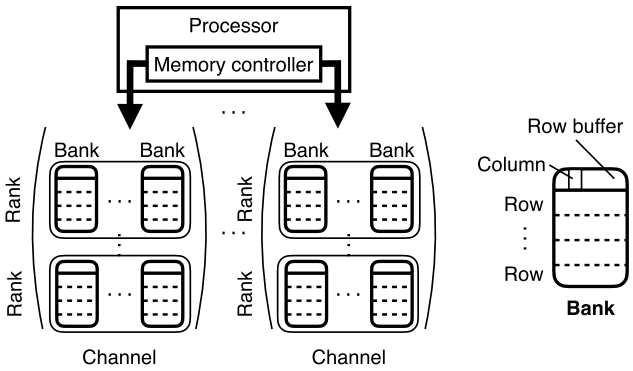


Fig. 1 Conceptual structure of multi-bank architectures.

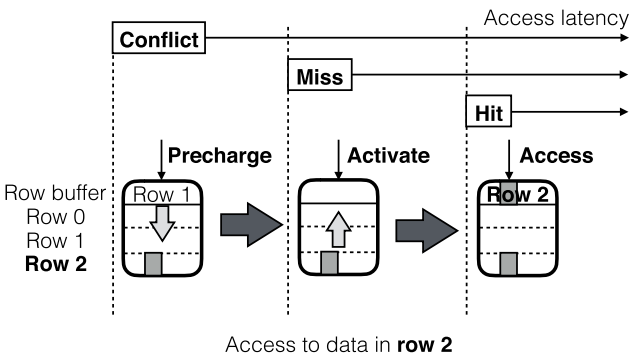


Fig. 2 Three types of row buffer accesses. A row buffer hit can skip precharge and activate commands, and a row buffer miss can skip a precharge command.

defined as the ratio of row buffer hits to all memory accesses. Since the latency of a memory access becomes longer in the order of a row buffer hit, miss, and conflict, improving RBL leads to a performance improvement.

There are two well-known policies to manage row buffers: *open-page* and *closed-page*. The former keeps a row in a row buffer after it is accessed. While subsequent accesses to the same row become row buffer hits, accesses to different rows become row buffer conflicts. On the other hand, the latter writes back a row in a row buffer to the original row immediately after it is accessed. Thus, all memory accesses become row buffer misses. In order to leverage the advantages of both policies, modern memory controllers apply adaptive open-page policies. For example, the memory controllers of AMD Bulldozer processors keep a row buffer open during 128 memory clocks after it is accessed and close it if there is no access during this period [17].

Address mapping schemes of memory controllers determine the location of data in DRAM using a physical address and significantly affect the DRAM energy consumption. Park et al. developed a tool to analyze address mapping schemes on real platforms and revealed that contiguous cache line-sized (64 B) blocks are interleaved across multiple channels on an Intel Nehalem platform [12]. We call this the *per-cache-line channel interleaving* (PCL) scheme. This scheme has been commonly used as a baseline in prior

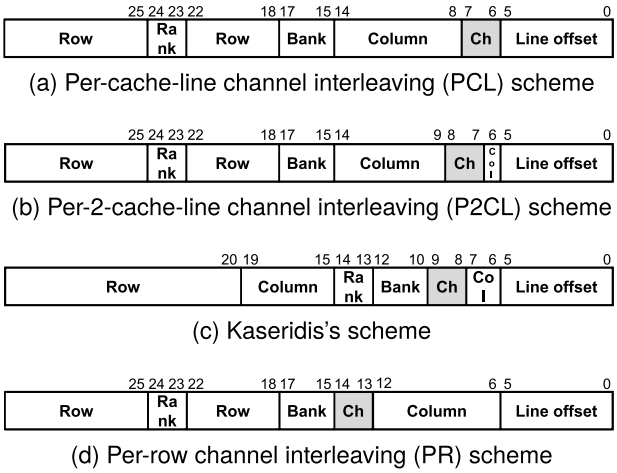


Fig. 3 Physical addresses of four address mapping schemes. Gray portions represent the channel ID bits.

Table 1 Simulation parameters

	Parameters
Processor	16 out-of-order cores, 3.0 GHz 4 issue width, 5 commit width
L1 I cache	Private, 32 KB, 4-way, 2-cycle latency
L1 D cache	Private, 32 KB, 8-way, 4-cycle latency
L2 cache	Private, 256 KB, 8-way, 6-cycle latency
L3 cache	Shared, 16 MB, 16-way, 30-cycle latency
Memory controller	Adaptive open-page policy [17] FR-FCFS scheduling policy [5]
DRAM	32 GB, DDR3-1333, 1 KB row 4 channels, 4 ranks/channel, 8 banks/rank
Timing	tCL=10, tRCD=10, tRP=10, tRAS=24
Current (mA)	IDD0=130, IDD1=155, IDD2P=10, IDD2N=70, IDD3P=60, IDD3N=90, IDD4W=300, IDD4R=255, IDD5=305, IDD6=9
Voltage (V)	VDD=1.5

work [4], [9], [10]. Figure 3 (a) illustrates a physical address for the PCL scheme with DRAM parameters summarized in Table 1. In this case, the 2-bit channel ID is located at the 6th bit. We also investigate schemes used on Intel SandyBridge and Haswell platforms with the analysis tool and observe that contiguous two cache line-sized (128 B) blocks are interleaved across channels. We call this the *per-2-cache-line channel interleaving* (P2CL) scheme. As shown in Fig. 3 (b), the channel ID is located at the 7th bit by inserting one bit of the column ID at the right side of the channel ID. Moreover, Kaseridis et al. proposed a scheme that interleaves contiguous four cache line-sized (256 B) blocks across channels, banks, and ranks [4]. Figure 3 (c) shows that the channel ID is located at the 8th bit. This scheme aims to achieve high BLP while retaining RBL within each 256 B block. Finally, some literature mentions a scheme that interleaves contiguous row-sized blocks across channels [8], [15]. We call this the *per-row channel interleaving* (PR) scheme, which locates the channel ID at the 13th bit as shown in Fig. 3 (d).

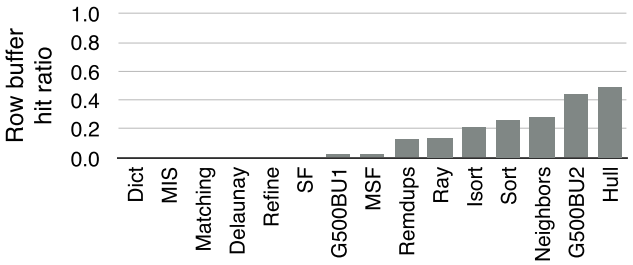


Fig. 4 Row buffer hit ratio of 15 multithreaded benchmarks on a real sever that applies the P2CL scheme.

2.2 Evaluation of Row Buffer Locality on a Real Server

We evaluate the row buffer hit ratio (i.e., RBL) of 15 multithreaded benchmarks on a real server that applies the P2CL scheme. The details of the experimental setup are explained in Sect. 4.2 and Sect. 6.1. Figure 4 demonstrates that RBL is relatively high for only a few benchmarks such as G500BU2 and Hull, but is generally low across the other benchmarks. It is an effective approach for reducing the DRAM energy consumption to improve low RBL.

3. Related Work

In this section, we introduce prior work that aims to efficiently leverage row buffers and/or BLP.

Improving the efficiency of row buffers: Zhang et al. proposed an address mapping scheme to reduce row buffer conflicts with retaining the RBL of single-threaded programs [3]. Their scheme arranges data causing frequent row buffer conflicts onto different banks by xor-ing two different parts of a physical address. As they assume single-channel memory systems, it does not apply channel interleaving. Memory access scheduling of memory controllers has also been thoroughly studied. The *FR-FCFS* policy, which is widely used in modern memory controllers, improves RBL by providing memory accesses to opened row buffers with high priorities [5]. Since this policy may cause unfair scheduling when multiple processes are executed simultaneously, Mutlu et al. [6] and Kim et al. [7] devised fairness-aware scheduling polices for multi-programmed workloads. Sudan et al. proposed techniques that co-locate heavily referenced cache blocks from different OS pages in a single row buffer to improve RBL [18]. Muralidhara et al. proposed a channel partitioning scheme that partitions data with different characteristics onto separate channels in order to avoid row buffer conflicts [8]. Park et al. observed that the similar memory access patterns of multiple threads cause frequent row buffer conflicts and introduced randomness in an OS page allocator to scatter memory accesses of multiple threads over different banks [12], [19]. Liu et al. implemented an OS-based bank partitioning technique that allocates distinct banks to each thread in order to eliminate row buffer conflicts among multiple threads [9].

Improving BLP: As the above bank partitioning

technique limits BLP for each thread, Jeong et al. [10] and Xie et al. [11] proposed new bank partition techniques that balance RBL and BLP. In order to improve BLP for each thread, Jeong's technique divides a rank into multiple sub-ranks, and Xie's technique allocates the appropriate number of banks to each thread in response to its BLP requirement. Tang et al. showed that multithreaded programs with irregular memory access patterns exhibit very low BLP and presented a compiler/runtime-based loop iteration scheduling strategy to maximize BLP [13]. Kaseridis et al. proposed the address mapping scheme described in Sect. 2.1 to improve RBL with sustaining high BLP [4]. They also devised a mechanism that combines open-page and closed-page policies using prefetch engines to improve RBL.

In this paper, we focus on address mapping schemes applying channel interleaving, because they significantly affect both RBL and BLP. We compare the four schemes shown in Fig. 3 and show that the PR scheme achieves the lowest DRAM energy consumption for multithreaded programs. Although various state-of-the-art schemes were introduced [20], [21], we focus on evaluating how the granularity of channel interleaving affects the DRAM energy consumption.

4. Memory Access Analysis

In this section, we analyze the memory access patterns of representative benchmarks with the PCL scheme on a simulator and show how the PR scheme can improve the DRAM energy consumption.

4.1 Simulator Setup

We use a cycle-accurate multicore simulator MARSSx86 [22] combined with a DRAM simulator DRAMSim2 [23]. As summarized in Table 1, a 3.0 GHz 16-core processor with 32 GB of DRAM is configured. The cache-line size of each cache memory is 64 B. The row buffer management policy is the adaptive open-page policy of AMD Bulldozer processors [17] explained in Sect. 2.1. Row buffers are also closed after accessed four times in order to prevent bank starvation, which is the default setting of DRAMSim2. DRAM contains four channels, four ranks per channel, and eight banks per rank. Thus, the total number of banks is 128. For timing, current, and voltage parameters of DRAM, we use a configuration file included in DRAMSim2 which assumes a DDR3-1333 DIMM. Note that we do not implement hardware prefetchers in this simulator, but address mapping schemes of memory controllers affect memory accesses issued by prefetchers.

4.2 Multithreaded Benchmarks

We use 13 multithreaded benchmarks from the problem-based benchmark suite (PBS) [24] and the Graph500 benchmark that executes breadth-first search on a graph [25]†. The BFS benchmark in PBS that also executes breadth-first search is substituted by the Graph500 benchmark. Table 2 describes these benchmarks. For the Graph500 benchmark, we use a state-of-the-art multithreaded implementation that consists of *top-down* and *bottom-up* algorithms [26]. Note that the latter is only used in our evaluation because it spends almost the whole execution time. Moreover, as it consists of two portions with different characteristics, we separately evaluate them (labeled as G500BU1 and G500BU2). All of the 15 benchmarks are compiled by GCC 4.8.5 with OpenMP support and executed with 16 threads.

Table 2 Descriptions of 15 multithreaded benchmarks

Type	Benchmark	Input data
Sequences	Sort	2^{24} random floating points
	Isort Remdups Dict	2^{24} random unsigned integers
Graphs	G500BU1 G500BU2	Kronecker graph with 2^{22} vertices and 2^{26} edges
	MIS Matching MSF SF	R-MAT graph with 2^{22} vertices and 2^{26} edges
Geometry & graphics	Hull	2^{24} random points within a unit circle
	Neighbors	2^{24} 2D random points in a cube
	Delaunay Refine	2^{24} random points a unit circle
	Ray	Happy Buddha

4.3 Memory Access Traces with the PCL Scheme

With the memory access traces of the 15 benchmarks using the PCL scheme, we observe that they are classified into two groups: *high-RBL* and *low-RBL*. The G500BU2 benchmark is a representative of the former group, and Figs. 5 (a) and 5 (b) show its results with 16 threads and one thread, respectively. Note that the other threads have similar access patterns. For this benchmark, we obtain four observations as follows.

1. In Fig. 5 (a), almost all accesses follow activate commands (•), and each bank is accessed only once until its row buffer is closed by a precharge command (■). This implies that these accesses are row buffer misses.
2. In Fig. 5 (a), almost all banks are not conflicted by multiple threads. This is because each thread accesses its private data, which is larger than the row size and is interleaved across different banks.
3. In Fig. 5 (b), a certain thread has a regular memory access pattern. Consecutive accesses are distributed to four banks that are 32 banks apart, which implies that this thread sequentially accesses contiguous 64 B blocks interleaved across four channels by the PCL

†The SA and Nbody benchmarks in PBS are omitted because they cannot be executed in our experimental environment due to segmentation faults.

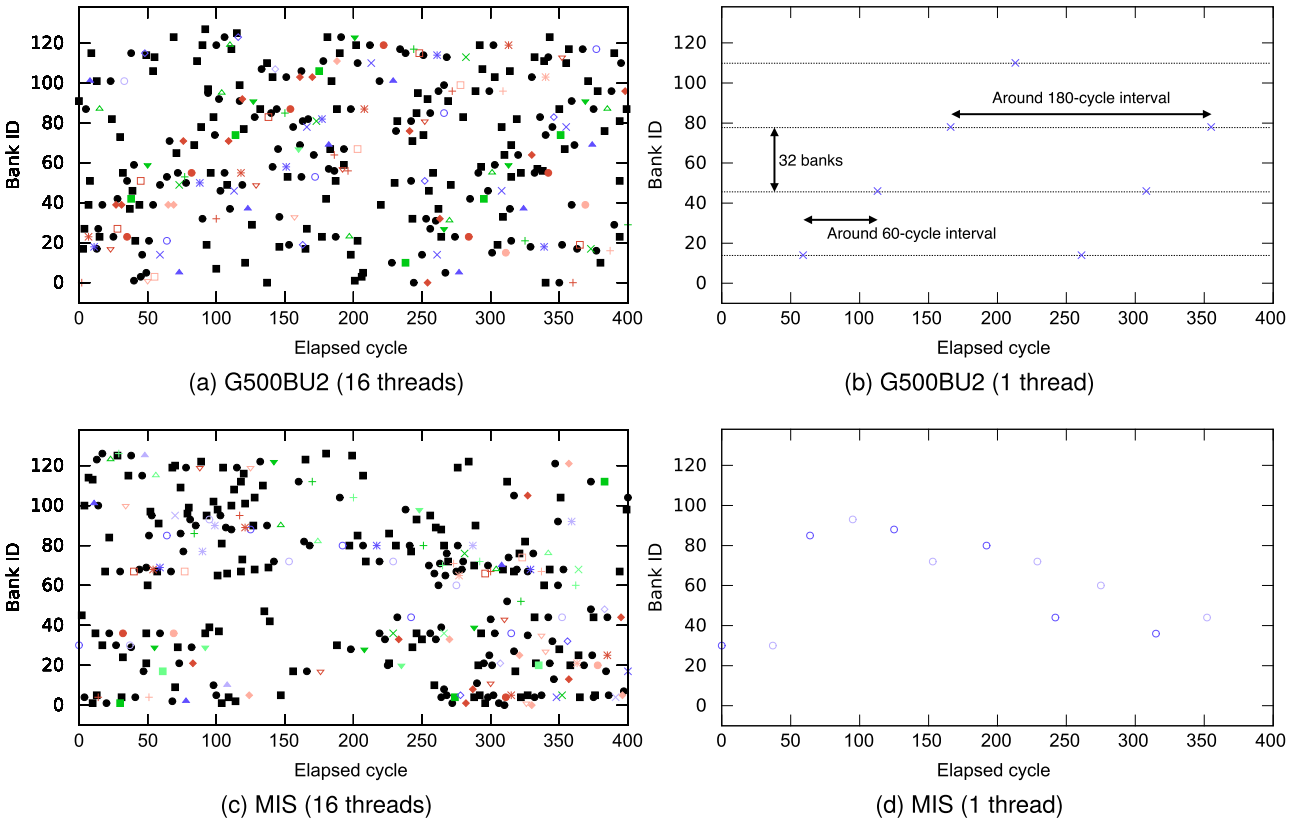


Fig. 5 Memory access traces with the PCL scheme. The dots are classified into three types. Black circles (●) and squares (■) correspond to activate and precharge commands, respectively. The colored dots with different colors and shapes indicate memory accesses requested by different threads.

scheme.

4. In Fig. 5 (b), consecutive accesses reach to DRAM at a 60-cycle interval, which means that BLP is not exploited within each thread. Moreover, a particular bank is accessed at a 180-cycle interval because it is accessed once every four accesses. Therefore, a row buffer of each bank is closed during this interval by the adaptive open-page policy.

From the above observations, we figure out that (1) the PCL scheme hurts the RBL of each thread by interleaving its private data across multiple banks of different channels, and (2) BLP is exploited across multiple threads, but not within each thread.

Next, Figs. 5 (c) and 5 (d) show the results of the MIS benchmark which is a representative of the low-RBL group. In Fig. 5 (c), almost all accesses are row buffer misses in similar to Fig. 5 (a), but the reason is different. Figure 5 (d) demonstrates that each thread has an irregular memory access pattern, and a particular bank is not accessed consecutively.

4.4 Memory Access Traces with the PR Scheme

With the observations in the previous section, we can expect that the RBL of each thread will be improved if each thread sequentially accesses a row in a particular bank. In this case, although each thread cannot access multiple banks in parallel, high BLP can be sustained because different threads will access different banks in parallel. Thus, we here focus on the PR scheme. Since it interleaves a contiguous physical address space across channels in the granularity of rows, each thread can sequentially access a whole row.

Figure 6 plots the memory access traces of the G500BU2 benchmark with the PR scheme. As expected, each thread consecutively accesses the same bank, and different threads access different banks in parallel. Compared to the results with the PCL scheme (Fig. 5 (a)), the PR scheme obviously improves the RBL of each thread and reduces the numbers of activate (●) and precharge (■) commands. Thus, the performance of this benchmark should be improved. Moreover, as the number of banks used in parallel is reduced, the power consumption of DRAM will be reduced. Current DRAM devices become a low-power state when all banks in a rank are closed [27]. The smaller number of banks used in parallel will increase this opportunity.

Although the PR scheme is beneficial to the memory access patterns as shown in Fig. 5 (a), it may significantly hurt performance if multiple threads cannot exploit BLP (e.g., for single-thread programs). In this case, fine-grained channel interleaving schemes such as the PCL scheme will outperform the PR scheme by exploiting BLP within a thread across multiple channels. Therefore, we hope that

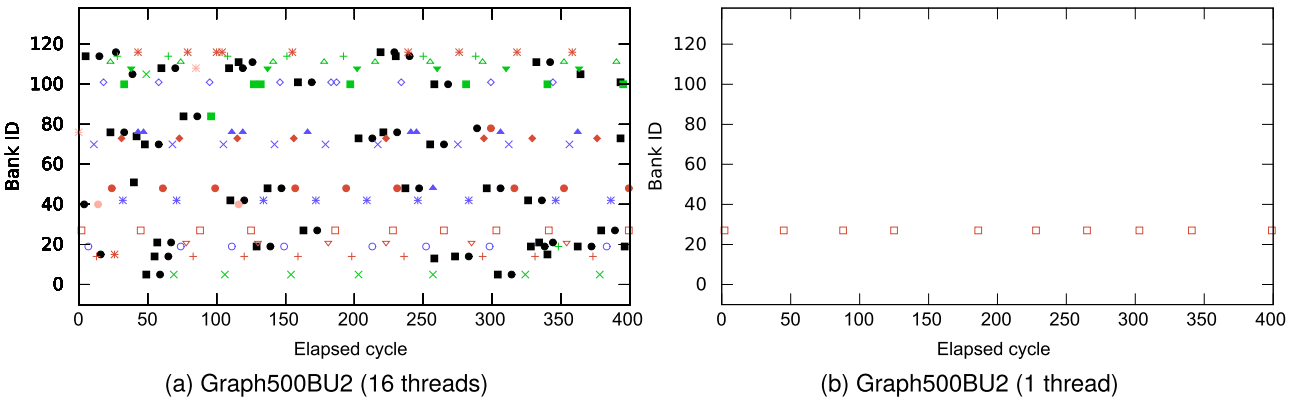


Fig. 6 Memory access traces with the per-row channel interleaving (PR) scheme.

we can change different channel interleaving schemes on future server platforms. Some servers have a BIOS option to enable or disable channel interleaving. Although disabling channel interleaving can improve the RBL of each thread in similar to the PR scheme, the bandwidth of multiple channels cannot be fully exploited because a single channel is preferably used.

5. Evaluation of DRAM Energy Consumption

In this section, we evaluate the DRAM energy consumption on the simulator with the four address mapping schemes. We first analyze results with a certain simulator setup in detail. Second, we conduct a sensitivity analysis in terms of the numbers of cores and channels. Finally, we discuss the results in consideration of DDR4.

5.1 Evaluation with 16 Cores and 4 Channels

Each of the 15 benchmarks is executed during a hundred million instructions on the simulator with parameters summarized in Table 1. Figure 7 shows row buffer hit ratio (i.e., RBL), the number of cycles, and the DRAM energy consumption with the four address mapping schemes. The DRAM energy consumption is calculated as the number of cycles times the average power consumption of DRAM obtained from the DRAMSim2 simulator.

As shown in the top figure, the ten benchmarks at the left side (Hull to G500BU1) are classified as the high-RBL group. In this group, RBL is improved in the order of PCL, P2CL, Kaseridis, and PR, because the number of consecutive accesses to the same row is increased in this order. Exceptionally, Kaseridis's scheme degrades the RBL of the Refine and G500BU1 benchmarks compared to the P2CL scheme. Since Kaseridis's scheme interleaves contiguous 256 B blocks across channels, ranks, and banks, the number of banks each thread accesses in parallel becomes larger than the other schemes, which causes a larger number of row buffer conflicts among multiple threads. In contrast, the PR scheme achieves the highest RBL for all of the benchmarks. The five benchmarks at the right side (Ray to Matching) originally expose low RBL, but the PR scheme can efficiently exploit their low RBL.

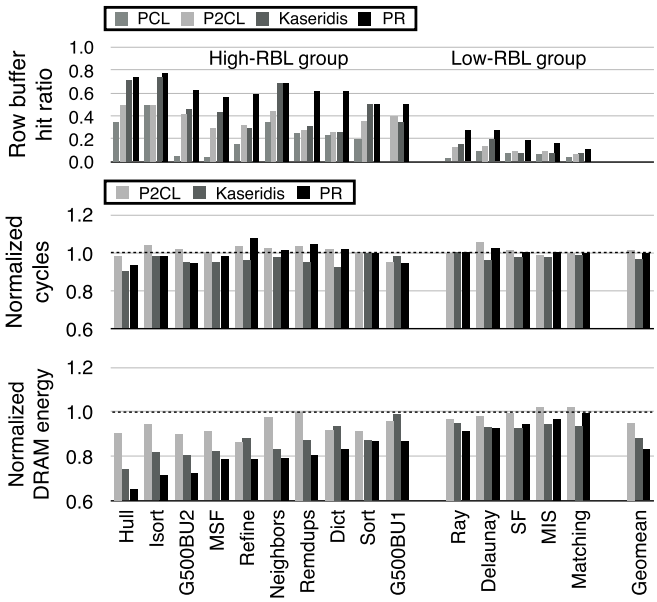


Fig. 7 Evaluation results of four address mapping schemes with 16 cores and four channels. The number of cycles and DRAM energy consumption are normalized by the results with the PCL scheme.

In terms of the number of cycles shown in the middle figure, there is a trade-off between RBL and BLP for the high-RBL group. As the number of banks each thread accesses in parallel is increased, BLP is improved but the RBL of each thread is decreased, and vice versa. Since the PCL scheme interleaves contiguous 64 B blocks across four channels, each thread accesses four banks in parallel. The P2CL scheme can improve RBL by arranging two contiguous 64 B blocks in the same row, whereas BLP cannot be exploited between these blocks. Consequently, the P2CL scheme scarcely reduces the number of cycles. Kaseridis's scheme can further improve RBL by arranging four contiguous 64 B blocks in the same row with sustaining high BLP by interleaving clusters of the four blocks across channels, ranks, and banks. Thus, this scheme reduces the number

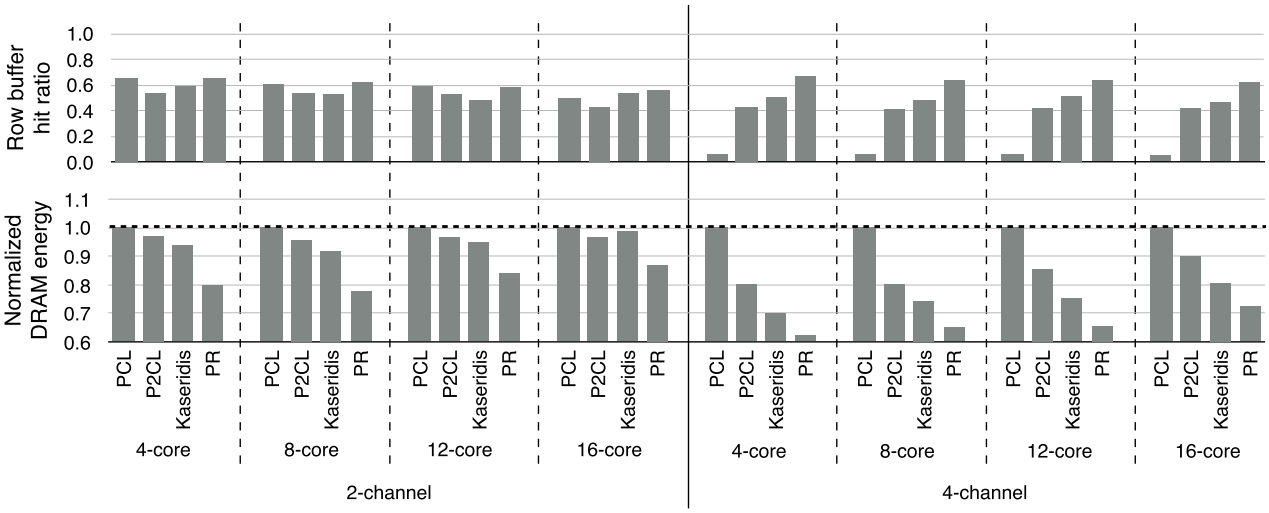


Fig. 8 Evaluation of four address mapping schemes with various numbers of cores and channels for the G500BU2 benchmark.

of cycles for some benchmarks such as Hull and Dict. The PR scheme also reduces the number of cycles for the Hull, G500BU2, and G500BU1 benchmarks, by improving RBL within each thread with sustaining high BLP across multiple threads. However, it has negative effects for the Refine and Remdups benchmarks, because each thread of them can efficiently exploit BLP. On the other hand, the five benchmarks in the low-RBL group are not largely affected by the address mapping schemes. The rightmost bars indicate the geometric means across the 15 benchmarks, which shows that Kaseridis's scheme slightly outperforms the other schemes.

The bottom figure shows that the DRAM energy consumption is generally reduced in the order of PCL, P2CL, Kaseridis, and PR for the high-RBL group. This is because the DRAM power consumption is reduced in this order due to a decreasing number of banks accessed in parallel. Modern DRAM devices enter power-down modes when all banks in a rank are closed, and the DRAMSim2 simulator models this functionality. The smaller number of banks accessed in parallel can increase this opportunity. However, Kaseridis's scheme consumes higher energy than the P2CL scheme for several benchmarks, because it tends to access more banks in parallel. In contrast, the PR scheme achieves the lowest energy consumption for all of the benchmarks in the high-RBL group. Moreover, it is also beneficial to the benchmarks in the low-RBL group. It reduces the DRAM energy consumption on average by 16.7%, 12.3%, and 5.5% (up to 34.7%, 28.2%, and 12.0% for the Hull benchmark) compared to the PCL, P2CL, and Kaseridis schemes, respectively.

5.2 Sensitivity Analysis

In addition, as the memory access patterns of multithreaded programs may depend on the numbers of cores and channels, we evaluate the four address mapping schemes with varying them on the simulator. Modern processors contain multiple memory controllers per chip, each of which manages different numbers of cores and channels [28]. For example, two memory controllers on a 12-core processor manage a group of eight cores and that of four cores, respectively. Moreover, high-end processors commonly include two to four channels. Thus, this sensitivity analysis is important to validate the benefit of the PR scheme for various configurations of memory controllers. We change the number of cores to 4, 8, 12, and 16 and the number of channels to 2 and 4. The other parameters are same as those in Table 1. Figure 8 shows the row buffer hit ratio (i.e., RBL) and DRAM energy consumption of the G500BU2 benchmark with the four schemes. Note that the DRAM energy consumption is normalized by the result of the PCL scheme with each configuration.

Row buffer hit ratio is obviously improved in the order of PCL, P2CL, Kaseridis, and PR with four channels, but not with two channels. In the case of the PCL scheme, each thread accesses the same bank once every four times with four channels, whereas it does so once every two times with two channels. Thus, an interval between two consecutive accesses to the same bank is halved with two channels. For example, a 180-cycle interval in Fig. 5 (b) becomes around 90 cycles. Since this interval is shorter than the duration to close row buffers (128 cycles), the second access becomes a row buffer hit. The PCL scheme, therefore, achieves over 50% of row buffer hit ratio with two channels, and the impacts of the other schemes become small. However, the PR scheme reduces the DRAM energy consumption in some extent by reducing the DRAM power consumption.

On the other hand, the number of cores scarcely affects RBL and the DRAM energy consumption. Intuitively, an interval between consecutive accesses to the same bank may become shorter as the number of cores is decreased, because the number of memory accesses a memory controller must schedule in a time unit is reduced. If the interval becomes shorter than 128 cycles, row buffer hit ratio will be

Table 3 Advantages of DDR4 over DDR3 [29]

Feature	DDR3	DDR4
Data rate [Mb/s]	800-2133	1600-3200
Densities	512Mb-8Gb	2Gb-16Gb
Internal banks	8	16
Voltage	1.5 V	1.2 V

improved. However, the above observation implies that the interval is independent of the number of cores, and it is determined by instruction counts executed by each thread between consecutive accesses. We investigate how the length of the interval affects the benefit of the PR scheme in the next section.

In summary, Fig. 8 demonstrates that the PR scheme is beneficial to various configurations of memory controllers. Compared to Kaseridis's scheme, it reduces the DRAM energy consumption by 11.2% to 15.1% with two channels and 10.6% to 12.8% with four channels, respectively.

5.3 Discussion on the Assumption of DDR4

Although we have evaluated four address mapping schemes on the assumption of DDR3, recent servers typically apply DDR4. Its main advantages over DDR3 are a higher data rate, higher density with more internal banks, and lower voltage, as summarized in Table 3. With a lower voltage, DDR4 DRAM can reduce power consumption by 20–30% compared to the same size of DDR3 DRAM [30], [31]. However, the system-level DRAM power consumption may be increased by migrating from DDR3 to DDR4, because the maximum DRAM capacity on a single server will be doubled. In this case, the amount of power reduction by the PR scheme will be larger than our results with DDR3. On the other hand, there is not a large difference in access latency between DDR3 and DDR4 devices. For example, tCL, tRCD, and tRP are 13-14 ns on a DDR3 DIMM [32] and 13-15 ns on a DDR4 DIMM [33], respectively. Thus, the impact of the PR scheme on performance is not largely changed by migrating from DDR3 to DDR4. Consequently, the amount of energy reduction by the PR scheme will be larger on a server with DDR4 compared to our results with DDR3.

6. Emulation of the PR Scheme on a Real Server

In this section, we emulate the PR scheme using a micro-benchmark on a real server and investigate its impacts on performance and the DRAM energy consumption. We describe the experimental setup and then show evaluation results.

6.1 Experimental Setup

We use a server that contains an 8-core (16-thread) Intel Xeon E5-4640 processor and four-channel 128 GB DDR3. On this server, we can enable or disable channel interleaving via a BIOS setting. When channel interleaving is enabled, the P2CL scheme is applied. The power consumption of DRAM is obtained via Intel Running Average Power Limit (RAPL) [34]. Moreover, the numbers of memory accesses and row buffer hits are obtained from hardware performance counters with an open-source tool called Likwid [35].

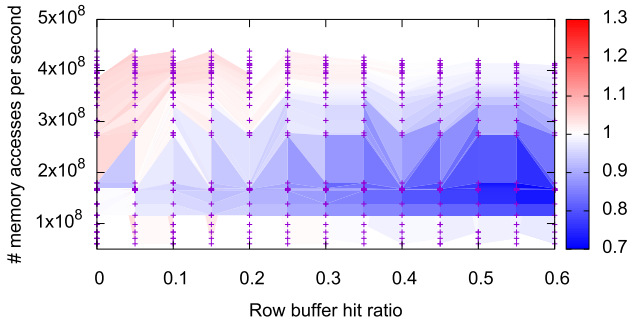
Algorithm 1 Pseudo code of our micro-benchmark

```
Input: RBHitRatio(%), numInc
Allocates 128/#threads GB of integers and initializes them to 0
for i ← 1 to 1,000,000 do
    for j ← 1 to RBHitRatio do   # Hit loop
        tmp ← the first integer in a cache-line-sized block
        for k ← 1 to numInc do
            tmp++
        end for
        Select the next cache-line-sized block
    end for
    for j ← 1 to (100 - RBHitRatio) do   # Miss loop
        tmp ← the first integer in a row-sized block
        for k ← 1 to numInc do
            tmp++
        end for
        Select the next row-sized block
    end for
end for
```

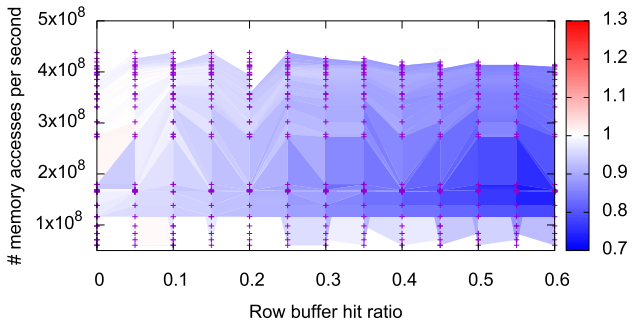
We emulate the PR scheme using a multithreaded micro-benchmark with channel interleaving disabled. Each thread of this benchmark executes the pseudo code shown in Algorithm 1 in parallel. It can adjust row buffer hit ratio (i.e., RBL) and an interval between memory accesses with two input parameters: *RBHitRatio* and *numInc*. The former represents the parentage of row buffer hit ratio from 0 to 100, and the latter is the number of increment operations executed between memory accesses. Each thread first allocates and initializes its private data so that the total data size of all threads fits 128 GB memory. As each private data is stored on different banks across four channels, this benchmark can exploit BLP among multiple threads. After that, the following two loops are repeated a million times. In the first loop called *hit loop*, each thread accesses contiguous cache line-sized blocks as many times as the RBHitRatio parameter. As we disable channel interleaving and software/hardware prefetchers for this benchmark, each thread sequentially accesses an entire row on DRAM, and these accesses become row buffer hits. In each iteration of the hit loop, the first integer in the current block is incremented as many times as the numInc parameter. The duration of these increments determines an interval between memory accesses. On the other hand, in the second loop called *miss loop*, each thread accesses contiguous row-sized blocks as many times as (100–RBHitRatio) and increments the first integer of each block in similar to the hit loop. Since different rows are accessed in each iteration, these accesses become row buffer misses.

6.2 Evaluation Results

We run the above benchmark with 16 threads on the experimental server. We evaluate its execution time and DRAM energy consumption by comparing the emulated PR scheme



(a) Normalized execution time



(b) Normalized DRAM energy

Fig. 9 Comparison of the emulated PR scheme with the P2CL scheme on a real server platform.

with the P2CL scheme as a baseline. For this baseline, we enable channel interleaving and always set the RBHitRatio parameter to zero. That is, all memory accesses become row buffer misses with the P2CL scheme. Figure 9 plots the results with the emulated PR scheme which are normalized by those with the P2CL scheme. The x-axis represents row buffer hit ratio improved by the PR scheme, which corresponds to the RBL each thread potentially possesses. The y-axis shows the number of memory accesses per second (i.e., the memory access frequency of each thread). Deeper red and blue colors mean a larger increase and reduction in each metric, respectively.

In the top figure, we can see that the execution time is reduced more significantly as the RBL becomes higher. Moreover, the impact of the PR scheme strongly depends on the frequency of memory accesses. If the frequency is high, the PR scheme hurts BLP each thread can exploit between consecutive accesses. On the other hand, if the frequency is low, the impact of the PR scheme is small because each thread is not memory-intensive. Consequently, the PR scheme degrades performance if RBL is low and the frequency of memory accesses is high (at the upper left in the figure). In contrast, it improves performance if RBL is high and the frequency of memory accesses is moderate (at the middle right in the figure). In the bottom figure, we observe that the PR scheme can reduce the DRAM energy consumption in various situations. This is because it can reduce the DRAM power consumption by reducing the number of banks used in parallel. Even if it degrades performance, it has almost no negative impact on the DRAM energy consumption.

7. Conclusions

We investigate the memory access patterns of multithreaded benchmarks using a simulator and reveal the reason why the RBL of them is low. With our observations, we focus on the PR scheme and demonstrate that it can improve the RBL of each thread with sustaining high BLP across multiple threads. Our evaluation on the simulator shows that it reduces the DRAM energy consumption by 16.7%, 12.3%, and 5.5% on average (up to 34.7%, 28.2%, and 12.0%) compared to three cache line-grained schemes, respectively. Moreover, our sensitivity analysis shows that it is beneficial to various configurations of memory controllers. Finally, we emulate the PR scheme on a real server and observe that its impact depends on the RBL and memory access frequency of an executed program, but it can reduce the DRAM energy consumption with various situations.

Acknowledgments

This research project was supported by the Japan Science and Technology Agency (JST), the Core Research of Evolutionary Science and Technology (CREST).

References

[1] Intel, “Intel® Xeon® Processor E7-8890 v4.” https://ark.intel.com/products/93790/Intel-Xeon-Processor-E7-8890-v4-60M-Cache-2_20-GHz. Last accessed: March 5, 2018.
[2] K. Kumar, K. Doshi, M. Dimitrov, and Y.-H. Lu, “Memory Energy Management for an Enterprise Decision Support System,” ISLPED ’11, pp.277–282, 2011.
[3] Z. Zhang, Z. Zhu, and X. Zhang, “A Permutation-based Page Interleaving Scheme to Reduce Row-buffer Conflicts and Exploit Data Locality,” MICRO-33, pp.32–41, 2000.
[4] D. Kaseridis, J. Stuecheli, and L.K. John, “Minimalist Open-page: A DRAM Page-mode Scheduling Policy for the Many-core Era,” MICRO-44, pp.24–35, 2011.
[5] S. Rixner, W.J. Dally, U.J. Kapasi, P. Mattson, and J.D. Owens, “Memory Access Scheduling,” ISCA ’00, pp.128–138, 2000.
[6] O. Mutlu and T. Moscibroda, “Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors,” MICRO-40, pp.146–160, 2007.
[7] Y. Kim, M. Papamichael, O. Mutlu, and M. Harchol-Balter, “Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior,” MICRO-43, pp.65–76, 2010.
[8] S.P. Muralidhara, L. Subramanian, O. Mutlu, M. Kandemir, and T. Moscibroda, “Reducing Memory Interference in Multicore Systems via Application-aware Memory Channel Partitioning,” MICRO-44, pp.374–385, 2011.
[9] L. Liu, Z. Cui, M. Xing, Y. Bao, M. Chen, and C. Wu, “A Software Memory Partition Approach for Eliminating Bank-level Interference in Multicore Systems,” PACT ’12, pp.367–376, 2012.
[10] M.K. Jeong, D.H. Yoon, D. Sunwoo, M. Sullivan, I. Lee, and M. Erez, “Balancing DRAM locality and parallelism in shared memory CMP systems,” HPCA ’12, pp.1–12, 2012.
[11] M. Xie, D. Tong, K. Huang, and X. Cheng, “Improving system throughput and fairness simultaneously in shared memory CMP systems via Dynamic Bank Partitioning,” HPCA ’14, pp.344–355,

2014.
[12] H. Park, S. Baek, J. Choi, D. Lee, and S.H. Noh, "Regularities Considered Harmful: Forcing Randomness to Memory Accesses to Reduce Row Buffer Conflicts for Multi-core, Multi-bank Systems," ASPLOS '13, pp.181–192, 2013.
[13] X. Tang, M. Kandemir, P. Yedlapalli, and J. Kotra, "Improving Bank-Level Parallelism for Irregular Applications," MICRO-49, pp.1–12, 2016.
[14] B. Akin, F. Franchetti, and J.C. Hoe, "Understanding the Design Space of DRAM-Optimized Hardware FFT Accelerators," ASAP '14, pp.248–255, 2014.
[15] O. Mutlu, "Computer Architecture: Main Memory (Alternate Version)." http://slideplayer.com/slide/4744474/. Last accessed on March 5, 2018.
[16] S. Imamura, Y. Yasui, K. Inoue, T. Ono, H. Sasaki, and K. Fujisawa, "Power-Efficient Breadth-First Search with DRAM Row Buffer Locality-Aware Address Mapping," HPGDMP '16, pp.17–24, 2016.
[17] AMD, "BIOS and Kernel Developer's Guide (BKDG) for AMD Family 15h Models 00h-0Fh Processors," 2013. Rev 3.14.
[18] K. Sudan, N. Chatterjee, D. Nellans, M. Awasthi, R. Balasubramonian, and A. Davis, "Micro-pages: Increasing DRAM Efficiency with Locality-Aware Data Placement," ASPLOS '10, pp.219–230, 2010.
[19] D. Kang, H. Park, and J. Choi, "Effect of Page Frame Allocation Pattern on Bank Conflicts in Multi-core Systems," RACS '13, pp.467–472, 2013.
[20] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks," USENIX Security '16, pp.565–581, 2016.
[21] M. Jung, D.M. Mathew, C. Weis, N. Wehn, I. Heinrich, M.V. Natale, and S.O. Krumke, "ConGen: An Application Specific DRAM Memory Controller Generator," MEMSYS '16, pp.257–267, 2016.
[22] A. Patel, F. Afram, S. Chen, and K. Ghose, "MARSS: A Full System Simulator for Multicore x86 CPUs," DAC '11, pp.1050–1055, 2011.
[23] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator," Computer Architecture Letters, vol.10, no.1, pp.16–19, Jan. 2011.
[24] G.E. Blelloch, J.T. Fineman, P.B. Gibbons, and J. Shun, "Internally Deterministic Parallel Algorithms Can Be Fast," PPoPP '12, pp.181–192, 2012.
[25] R.C. Murphy, K.B. Wheeler, B.W. Barrett, and J.A. Ang, Introducing the Graph 500. Cray User's Group (CUG), 2010.
[26] Y. Yasui, K. Fujisawa, E.L. Goh, J. Baron, A. Sugiura, and T. Uchiyama, "NUMA-aware Scalable Graph Traversal on SGI UV Systems," HPGP '16, pp.19–26, 2016.
[27] Micron Technology, Inc., Calculating Memory System Power for DDR3, 2007.
[28] Intel, "Intel Xeon Processor E5 and E7 v3 Family Uncore Performance Monitoring Reference Manual," June 2015.
[29] Micron, "DDR4 - Advantages of Migrating from DDR3." https://www.micron.com/products/dram/ddr3-to-ddr4. Last accessed on March 7, 2018.
[30] Micron, "DRAM Memory In High-Speed Digital Designs." https://www.keysight.com/upload/cmc_upload/All/5Micron.pdf. Last accessed on March 1, 2018.
[31] C. Stephan, "Quantifying the Power Savings by Upgrading to DDR4 Memory on Lenovo Servers," LENOVO PRESS, 2016.
[32] Micron, "2Gb: x4, x8, x16 DDR3 SDRAM features," 2006.
[33] Micron, "4Gb: x4, x8, x16 DDR4 SDRAM features," 2014.
[34] H. David, E. Gorbatov, U.R. Hanebutte, R. Khanaa, and C. Le, "RAPL: Memory Power Estimation and Capping," ISLPED '10, pp.189–194, 2010.
[35] J. Treibig, G. Hager, and G. Wellein, "LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments," ICPPW '10, pp.207–216, 2010.

Satoshi Imamura received his B.E., M.E., and Ph.D. degrees in computer science from Department of Advanced Information Technology, Kyushu University in 2011, 2013, and 2017, respectively. He is currently a researcher at Fujitsu Laboratories Ltd. His research interests include architectures of microprocessors and memory for energy-efficient computing.

Yuichiro Yasui received his B.S. and M.E. degrees from Tokyo Denki University and Chuo University in 2007 and 2009, respectively. He currently works in Nikkei Business Publications, and is also a visiting research fellow at the Institute of Mathematics for Industry, Kyushu University. His research interests include energy-efficient NUMA-aware graph algorithms.

Koji Inoue received his B.E. and M.E. degrees in computer science from Kyushu Institute of Technology in 1994 and 1996, respectively. He received his Ph.D. degree in computer science from Kyushu University in 2001. In 1999, he joined Halo LSI Design & Technology, Inc., NY, as a circuit designer. He is currently a professor of the Department of I&E Visionaries, Kyushu University. His research interests include power-aware computing, high-performance computing, dependable processor architecture, secure computer systems, 3D microprocessor architectures, and multi/many-core architectures.

Takatsugu Ono received his Ph.D. degree in computer science from Kyushu University in 2009. In 2010, he joined Fujitsu Laboratories Ltd. as a researcher. He is currently an assistant professor at the Department of I&E Visionaries, Kyushu University. His research interests include architectures of emerging memory such as non-volatile memory, warehouse-scale computing, and secure computing. He is a member of the IEEE and the IPSJ.

Hiroshi Sasaki received his B.E., M.E., and Ph.D. degrees from the University of Tokyo in 2003, 2005, and 2008, respectively. He was a project assistant professor at the University of Tokyo from 2008 to 2011 and at Kyushu University from 2011 to 2014, and also a visiting researcher at IBM T.J. Watson Research Center from 2013 to 2014. He is currently a visiting researcher at Columbia University. His research interests include computer architecture and operating systems for microprocessors.

Katsuki Fujisawa received his B.S. and M.S. degrees in industrial engineering from Waseda University in 1993 and 1995, respectively, and his Ph.D. degree in mathematical and computing sciences from Tokyo Institute of Technology in 1998. He is currently a professor of Kyushu University. His research interests include mathematical optimization and high-performance computing. He is a member of SIAM, IPSJ, INFORMS.