Guest Editor Introduction: Recent Advances in Overcoming Bottlenecks in Memory Systems and Managing Memory Resources in GPU Systems

Onur Mutlu[1,2] Saugata Ghose[2] Rachata Ausavarungnirun[2]

[1]*ETH Zürich* [2]*Carnegie Mellon University*

Memory and storage systems are a fundamental system performance, energy, and reliability bottleneck in modern systems [5, 6, 7, 25, 28, 29]. This bottleneck is becoming increasingly severe due to (1) the very limited latency reductions in memory and storage devices over the last several years; (2) aggressive manufacturing process technology scaling and other techniques to improve memory density, such as multi-level cell technology, which increase the storage capacity of these devices, but introduce more raw bit errors and increase manufacturing process variation; (3) limited pin counts in chip packages, which prevent system designers from adding more and/or wider buses to increase bandwidth; (4) overwhelmingly data-intensive applications, which require high-bandwidth access to very large amounts of data; and (5) the increasing fraction of overall system energy consumed by memory systems and data movement. To make matters worse, it is becoming increasingly difficult to continue scaling these devices to smaller process technology nodes, and even though alternative emerging memory and storage technologies can potentially alleviate some of the shortcomings of existing memory and storage technologies, they also introduce *new* shortcomings that were previously absent. Therefore, there is a pressing need to comprehensively understand and mitigate these bottlenecks in both existing and emerging memory and storage systems and technologies.

This issue features extended summaries and retrospectives of some of the recent research done by our research group, SAFARI [33,39], on (1) various critical problems in memory systems and (2) how memory system bottlenecks affect graphics processing unit (GPU) systems. As more applications share a single system, operations from each application can contend with each other at various shared components within the system. If left unmitigated, such contention can undermine many of the benefits of parallelism, by slowing down each application or thread of execution [24, 26, 27, 28, 29]. The compound effect of contention, high memory latency and access overheads, as well as inefficient management of resources, greatly degrades performance, quality-of-service (QoS), and energy efficiency. The ten works featured in this issue study several aspects of (1) inter-application interference in multicore systems, heterogeneous systems, and GPUs; (2) the growing overheads and expenses associated with growing memory densities and latencies; and (3) performance, programmability, and portability issues in modern GPUs, especially those related to memory system resources.

These works rely on real system characterizations and simulation to develop a rigorous understanding of the interference and bottlenecks, and to provide solutions. Our analyses have shown key scaling and performance bottlenecks, proposed new solutions, and have inspired the research community to develop further investigations (e.g., on interference and fairness in main memory [41,42,43,45], subarray-level parallelism [8, 13], low-cost memory reliability [21], hybrid memory management [20,22,23,32,50]). In order to aid future research, we have released our flexible and extensible memory system simulator, Ramulator, as open-source software [15, 38], and have released open-source simulators that accurately model memory interference in multicore systems [34, 36] and memory resource bottlenecks in GPU systems [35, 37].

In each work that is featured in this issue, based on our rigorous studies and analyses, we propose novel solutions that mitigate many of these problems. We examine GPUs as a special example because they enable massively parallel processing on a single chip and, as a result, are limited greatly by the bottlenecks in the memory system. For each of the works presented in this special issue, its corresponding article examines the work's significance in the context of modern computer systems, and discusses several new research questions and directions that each work motivates.

We start with three of our works that manage interference and contention in main memory. When multiple applications (or multiple threads of one or more applications) concurrently issue memory requests, these requests often contend with each other in the main memory system, increasing the average memory access latency and reducing per-application or per-thread parallelism. This contention becomes especially problematic when a highly-memory-intensive application issues many more requests than other applications, causing requests from the other applications to unfairly wait for very long times as the memory system takes time to service all of the requests from the highly-memory-intensive application. To mitigate the interference that each application induces on the other applications, memory systems must adopt new mechanisms to regulate the available memory bandwidth among all applications and/or reduce the amount of memory-level contention. Doing so can enable systems that are higher performance, more predictable, and more energy efficient at the same time. The first three works featured in this issue enable new mechanisms to more efficiently manage interference and contention in main memory.

The first paper in the issue describes Memory Interference-induced Slowdown Estimation (MISE), which originally appeared in HPCA 2013 [43]. This work (1) develops a model called MISE, which predicts the impact of interference in DRAM on the overall system performance; and (2) uses this model to design new memory schedulers that improve fairness and QoS among concurrently-executing applications. The work finds that various MISE-based memory schedulers can (1) provide predictable performance to designated applications and (2) significantly improve the overall system throughput.

The second paper in the issue describes Staged Memory Scheduling, which originally appeared in ISCA 2012 [3]. This work analyzes the high impact of interference between the CPU and GPU in a heterogeneous system (e.g., a system-on-chip), showing that the GPU can overwhelm CPU performance and sometimes vice versa. Based on this finding, the work develops a new memory controller that provides fair memory access for both CPU and GPU applications, improving the performance of CPU applications without affecting the throughput of GPU applications.

The third paper in the issue describes Subarray-Level Parallelism (SALP), which originally appeared in ISCA 2012 [13]. This work exploits the subarrays (i.e., sub-banking) in DRAM architectures to greatly increase the amount of memory parallelism available to applications. SALP proposes three new mechanisms to expose the subarrays to the memory controller at low cost, improving row locality and reducing the number of high-latency bank conflicts that occur when multiple requests access the same memory bank. The reduced bank conflicts and the improved row locality significantly improve overall system performance and reduce energy consumption.

Next, we look at several of our works that address the growing overheads and expenses associated with growing main memory densities and latencies. As systems execute more applications in parallel, and as applications process larger amounts of data, DRAM manufacturers have relied on aggressive technology scaling to increase the density of each DRAM device. Unfortunately, such scaling has introduced a number of key challenges [25, 28, 29], which we methodically address in the next four works.

Our fourth paper in the issue describes DSARP, which originally appeared in HPCA 2014 [8]. This work explores how increasing memory density will cause DRAM refresh operations to become a bigger performance bottleneck, preventing the DRAM from effectively servicing outstanding memory requests with low latency. The work proposes new memory controller policies that almost completely eliminate the performance overhead of DRAM refresh by performing refresh operations in the background via low-cost changes to the DRAM architecture and the memory controller.

Our fifth paper in the issue describes ChargeCache, which originally appeared in HPCA 2016 [12]. This work finds that many applications must reopen memory rows soon after they are closed because of interference (i.e., bank conflicts), incurring a high access latency. ChargeCache is a new mechanism that takes advantage of the high charge held within a recently-closed row to reduce the access latency to such a row when it is accessed again soon in the future. The work shows that ChargeCache significantly improves the overall system performance and energy consumption.

Our sixth paper in the issue describes heterogeneous-reliability memory (HRM), which originally appeared in DSN 2014 [21]. This work demonstrates on real machines that many data center applications can tolerate errors in large regions of their memory address spaces without affecting correctness. The work uses this observation to lower the cost of memory subsystems for data centers, by introducing a new memory system framework, HRM, where the memory system consists of different modules with different types and amounts of error correction/detection capabilities. By employing many DRAM modules without error correction and intelligently mapping error-tolerant memory regions to these modules and error-vulnerable memory regions to DRAM modules with error correction, HRM significantly reduces the cost of a data center system, while still providing high overall reliability and availability.

Our seventh paper in the issue describes row buffer locality aware (RBLA) caching, which originally appeared in ICCD 2012 [50]. This work proposes a new technique to manage data placement in hybrid memory systems, which combine conventional DRAM with emerging memory technologies to provide the benefits of both in a scalable yet cost-effective manner. Exploiting the key observation that row buffer hits are of the same cost in both DRAM and emerging memory technologies, RBLA avoids migrating data from the emerging memory to conventional DRAM (and vice versa) when the migration would not yield a significant benefit, thereby preserving the precious DRAM space for data that really benefits from the low access latency of DRAM arrays. The work shows that RBLA improves both system performance and energy consumption as a result.

Finally, we examine how to manage memory resources within GPUs. For many general-purpose GPU (GPGPU) applications, programmers are responsible for explicitly managing all memory resources, including registers, by specifying in programs how much each application should get of each resource. Our solutions automatically manage these resources in both hardware and software, and sometimes cooperatively between the hardware and software, transparently to the programmer. The solutions lift the burden of resource management from the programmer, and improve the performance and efficiency of GPGPU applications.

Our eighth paper in the issue describes Zorua, which originally appeared in MICRO 2016 [46]. Current GPU systems require programmers to discover and explicitly specify the quantities of each resource that are assigned to a thread, in order to avoid significant performance penalties. This work pro-

poses a new resource virtualization mechanism for GPGPU applications, called Zorua, which can assign resources to each thread dynamically at runtime based on the thread's needs and the available resources in the GPU, with only annotations provided by the compiler. With its effective resource virtualization, Zorua improves (1) programmability, by removing the existing burden on programmers to tune the thread resource allocation; (2) portability, by removing the need to retune the resource allocation when an application tuned for one GPU architecture is executed on a different GPU architecture; and (3) performance, by ensuring the careful allocation and oversubscription of resources to best utilize the hardware.

Our ninth paper in the issue describes Memory Divergence Correction (MeDiC), which originally appeared in PACT 2015 [4], This work finds that different warps (i.e., groups of threads that execute in lockstep) exhibit different levels of memory divergence, where some, but not all, threads stall on long-latency memory accesses, which prevents forward progress for all threads in the warp. MeDiC consists of three new mechanisms that work together to optimize cache and memory resource management in a GPU, based on the divergence behavior of the warps belonging to an application. These three mechanisms provide significant performance improvements for GPGPU applications.

Our tenth paper in the issue describes Mosaic, which originally appeared in MICRO 2017 [1]. In contemporary GPUs, limited resources for memory virtualization can cause a single operation (e.g., an address translation that misses in the GPU's translation lookaside buffer) to often stall hundreds of threads for long latencies, leading to significant underutilization of the GPU. The memory virtualization bottleneck can be alleviated by changing the page size, but a major hurdle to this is the key trade-off between two costly operations: demand paging (which benefits from small page sizes) and address translation (which benefits from large page sizes). This work proposes a new hardware mechanism that takes advantage of GPGPU memory access patterns to enable the efficient support of multiple page sizes transparently to the programmer. By efficiently supporting multiple page sizes, Mosaic alleviates the high contention for memory virtualization resources, which in turn significantly improves the performance of GPGPU applications.

Throughout all of these works, we (1) identify various points of interference, contention, and resource bottlenecks in memory systems and GPUs; and (2) appropriately modify the systems to mitigate these issues at low cost and low overhead. These works improve the performance, fairness, energy consumption, and/or programmability of a system, and often improve scalability as more applications execute concurrently on the system. Even though the works presented are described in the context of DRAM, the dominant memory technology of today, we believe many of the basic ideas and concepts can be applied or adapted to emerging memory technologies [22], e.g., phase-change memory [17, 18, 19, 31, 48, 49, 51], STT-MRAM [11, 16, 30], and memristors/RRAM [10, 40, 47]. We hope that the works featured in this special issue inspire readers to explore other sources of interference, contention, performance, and programmability issues in modern systems, and to develop new solutions that can enable fair, high-performance, energy-efficient systems for the future.

Acknowledgments

The works featured in this issue, along with our related works that we reference in each featured work, are a result of the research done together with many students and collaborators over the course of the past 10+ years, whose contributions we acknowledge. In particular, we acknowledge and appreciate the dedicated effort of current and former students and postdocs in our research group, SAFARI [33, 39], who contributed to the ten featured works, including Kevin Chang, Rachael Harding, Hasan Hassan, Kevin Hsieh, Ben Jaiyen, Samira Khan, Yoongu Kim, Donghyuk Lee, Yixin Luo, Justin Meza, Gennady Pekhimenko, Vivek Seshadri, Ashish Shrestha, Lavanya Subramanian, Nandita Vijaykumar, and HanBin Yoon.

Aside from our featured works and other referenced papers from our group, where a wealth of information on modern memory and storage systems can be found, at least four Ph.D. dissertations have shaped the works that we feature in this special issue:

- Lavanya Subramanian's thesis entitled "Providing High and Controllable Performance in Multicore Systems Through Shared Resource Management" [44],
- Yoongu Kim's thesis entitled "Architectural Techniques to Enhance DRAM Scaling" [14],
- Kevin Chang's thesis entitled "Understanding and Improving the Latency of DRAM-Based Memory Systems" [9], and
- Rachata Ausavarungnirun's thesis entitled "Techniques for Shared Resource Management in Systems with Throughput Processors" [2].

We also acknowledge various funding agencies (the National Science Foundation, the Semiconductor Research Corporation, the Intel Science and Technology Center on Cloud Computing, CyLab, the CMU Data Storage Systems Center, and the NIH) and industrial partners (AMD, Facebook, Google, HP Labs, Huawei, IBM, Intel, Microsoft, NVIDIA, Oracle, Qualcomm, Rambus, Samsung, Seagate, VMware), and ETH Zürich, who have supported the featured works in this issue and other related work in our research group generously over the years.

References

[1] R. Ausavarungnirun, J. Landgraf, V. Miller, S. Ghose, J. Gandhi, C. J. Rossbach, and O. Mutlu, "Mosaic: A GPU Memory Manager with Application-Transparent Support for Multiple Page Sizes," in *MICRO*, 2017.

[2] R. Ausavarungnirun, "Techniques for Shared Resource Management in Systems with Throughput Processors," Ph.D. dissertation, Carnegie Mellon Univ., 2017.

[3] R. Ausavarungnirun, K. K. Chang, L. Subramanian, G. H. Loh, and O. Mutlu, "Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems," in *ISCA*, 2012.

[4] R. Ausavarungnirun, S. Ghose, O. Kayıran, G. H. Loh, C. R. Das, M. T. Kandemir, and O. Mutlu, "Exploiting Inter-Warp Heterogeneity to Improve GPGPU Performance," in *PACT*, 2015.

[5] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives," *Proc. IEEE*, 2017.

[6] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid-State Drives," arXiv:1706.08642 [cs.AR], 2017.

[7] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery," arXiv:1711.11427 [cs.AR], 2017.

[8] K. K. Chang, D. Lee, Z. Chishti, A. R. Alameldeen, C. Wilkerson, Y. Kim, and O. Mutlu, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," in *HPCA*, 2014.

[9] K. K. Chang, "Understanding and Improving the Latency of DRAM-Based Memory Systems," Ph.D. dissertation, Carnegie Mellon Univ., 2017.

[10] L. Chua, "Memristor—The Missing Circuit Element," *TCT*, 1971.

[11] X. Guo, E. Ipek, and T. Soyata, "Resistive Computation: Avoiding the Power Wall with Low-Leakage, STT-MRAM Based Computing," in *ISCA*, 2010.

[12] H. Hassan, G. Pekhimenko, N. Vijaykumar, V. Seshadri, D. Lee, O. Ergin, and O. Mutlu, "ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality," in *HPCA*, 2016.

[13] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," in *ISCA*, 2012.

[14] Y. Kim, "Architectural Techniques to Enhance DRAM Scaling," Ph.D. dissertation, Carnegie Mellon Univ., 2015.

[15] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A Fast and Extensible DRAM Simulator," *CAL*, 2015.

[16] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," in *ISPASS*, 2013.

[17] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *ISCA*, 2009.

[18] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Phase Change Memory Architecture and the Quest for Scalability," *CACM*, 2010.

[19] B. C. Lee, P. Zhou, J. Yang, Y. Zhang, B. Zhao, E. Ipek, O. Mutlu, and D. Burger, "Phase-Change Technology and the Future of Main Memory," *IEEE Micro*, 2010.

[20] Y. Li, S. Ghose, J. Choi, J. Sun, H. Wang, and O. Mutlu, "Utility-Based Hybrid Memory Management," in *CLUSTER*, 2017.

[21] Y. Luo, S. Govindan, B. Sharma, M. Santaniello, J. Meza, A. Kansal, J. Liu, B. Khessib, K. Vaid, and O. Mutlu, "Characterizing Application Memory Error Vulnerability to Optimize Datacenter Cost via Heterogeneous-Reliability Memory," in *DSN*, 2014.

[22] J. Meza, Y. Luo, S. Khan, J. Zhao, Y. Xie, and O. Mutlu, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," in *WEED*, 2013.

[23] J. Meza, J. Chang, H. Yoon, O. Mutlu, and P. Ranganathan, "Enabling Efficient and Scalable Hybrid Memories Using Fine-Granularity DRAM Cache Management," *CAL*, 2012.

[24] T. Moscibroda and O. Mutlu, "Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems," in *USENIX Security*, 2007.

[25] O. Mutlu, "The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser," in *DATE*, 2017.

[26] O. Mutlu and T. Moscibroda, "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors," in *MICRO*, 2007.

[27] O. Mutlu and T. Moscibroda, "Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems," in *ISCA*, 2008.

[28] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *IMW*, 2013.

[29] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems," *SUPERFRI*, 2014.

[30] H. Naeimi, C. Augustine, A. Raychowdhury, S.-L. Lu, and J. Tschanz, "STT-RAM Scaling and Retention Failure," *Intel Technology Journal*, 2013.

[31] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," in *ISCA*, 2009.

[32] J. Ren, J. Zhao, S. Khan, J. Choi, Y. Wu, and O. Mutlu, "ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems," in *MICRO*, 2015.

[33] SAFARI Research Group, http://www.ece.cmu.edu/~safari/.

[34] SAFARI Research Group, "ASMSim – GitHub Repository," https://github.com/CMU-SAFARI/ASMSim.

[35] SAFARI Research Group, "MeDiC GPGPU-Sim Patch – GitHub Repository," https://github.com/CMU-SAFARI/MeDiC.

[36] SAFARI Research Group, "MemSchedSim – GitHub Repository," https://github.com/CMU-SAFARI/MemSchedSim.

[37] SAFARI Research Group, "Mosaic Simulator – GitHub Repository," https://github.com/CMU-SAFARI/Mosaic.

[38] SAFARI Research Group, "Ramulator – GitHub Repository," https://github.com/CMU-SAFARI/ramulator.

[39] SAFARI Research Group, "SAFARI Software Tools – GitHub Repository," https://github.com/CMU-SAFARI/.

[40] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The Missing Memristor Found," *Nature*, 2008.

[41] L. Subramanian, D. Lee, V. Seshadri, H. Rastogi, and O. Mutlu, "The Blacklisting Memory Scheduler: Achieving High Performance and Fairness at Low Cost," in *ICCD*, 2014.

[42] L. Subramanian, D. Lee, V. Seshadri, H. Rastogi, and O. Mutlu, "BLISS: Balancing Performance, Fairness and Complexity in Memory Access Scheduling," *TPDS*, 2016.

[43] L. Subramanian, V. Seshadri, Y. Kim, B. Jaiyen, and O. Mutlu, "MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems," in *HPCA*, 2013.

[44] L. Subramanian, "Providing High and Controllable Performance in Multicore Systems Through Shared Resource Management," Ph.D. dissertation, Carnegie Mellon Univ., 2015.

[45] L. Subramanian, V. Seshadri, A. Ghosh, S. Khan, and O. Mutlu, "The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory," in *MICRO*, 2015.

[46] N. Vijaykumar, K. Hsieh, G. Pekhimenko, S. Khan, A. Shrestha, S. Ghose, A. Jog, P. B. Gibbons, and O. Mutlu, "Zorua: A Holistic Approach to Resource Virtualization in GPUs," in *MICRO*, 2016.

[47] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal-Oxide RRAM," *Proc. IEEE*, 2012.

[48] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase Change Memory," *Proc. IEEE*, 2010.

[49] H. Yoon, J. Meza, N. Muralimanohar, N. P. Jouppi, and O. Mutlu, "Efficient Data Mapping and Buffering Techniques for Multi-Level Cell Phase-Change Memories," *TACO*, 2014.

[50] H. Yoon, J. Meza, R. Ausavarungnirun, R. A. Harding, and O. Mutlu, "Row Buffer Locality Aware Caching Policies for Hybrid Memories," in *ICCD*, 2012.

[51] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," in *ISCA*, 2009.