Processing Data Where It Makes Sense in Modern Computing Systems: Enabling In-Memory Computation

Onur Mutlu
ETH Zürich and Carnegie Mellon University
omutlu@gmail.com

Abstract— **Today's systems are overwhelmingly designed to move data to computation. This design choice goes directly against at least three key trends in systems that cause performance, scalability and energy bottlenecks: 1) data access from memory is already a key bottleneck as applications become more data-intensive and memory bandwidth and energy do not scale well, 2) energy consumption is a key constraint in especially mobile and server systems, 3) data movement is very expensive in terms of bandwidth, energy and latency, much more so than computation.**

These trends are especially severely-felt in the data-intensive server and energy-constrained mobile systems of today. At the same time, conventional memory technology is facing many scaling challenges in terms of reliability, energy, and performance [1-23]. As a result, memory system architects are open to organizing memory in different ways and making it more intelligent, at the expense of slightly higher cost. The emergence of 3D-stacked memory plus logic as well as the adoption of error correcting codes inside the latest DRAM chips are an evidence of this trend.

In this talk, I will discuss some recent research that aims to practically enable computation close to data. After motivating trends in applications as well as technology, we will discuss at least two promising directions: 1) performing massively-parallel bulk operations in memory by exploiting the analog operational properties of DRAM, with low-cost changes [24-28], 2) exploiting the logic layer in 3D-stacked memory technology in various ways to accelerate important data-intensive applications [28-36]. In both approaches, we will discuss relevant cross-layer research, design, and adoption challenges in devices, architecture, systems, and programming models. Our focus will be the development of in-memory processing designs that can be adopted in real computing platforms at low cost.

Keywords-In-memory computation; DRAM; data movement; memory bottleneck

CURRICULUM VITAE

Onur Mutlu is a Professor of Computer Science at ETH Zurich. He is also a faculty member at Carnegie Mellon University, where he previously held the William D. and Nancy W. Strecker Early Career Professorship. His current broader research interests are in computer architecture, systems, and bioinformatics. He is especially interested in interactions across domains and between applications, system software, compilers, and microarchitecture, with a major current focus on memory and storage systems. A variety of techniques he, along with his group and collaborators, has invented over the years have influenced industry and have been employed in commercial microprocessors and memory/storage systems. He obtained his PhD and MS in ECE from the University of Texas at Austin and BS degrees in Computer Engineering and Psychology from the University of Michigan, Ann Arbor. His industrial experience spans starting the Computer Architecture Group at Microsoft Research (2006-2009), and various product and research positions at Intel Corporation, Advanced Micro Devices, VMware, and Google. He received the inaugural IEEE Computer Society Young Computer Architect Award, the inaugural Intel Early Career Faculty Award, faculty partnership awards from various companies, a healthy number of best paper or "Top Pick" paper recognitions at various computer systems and architecture venues, and the ACM Fellow recognition "for contributions to computer architecture research, especially in memory systems." His computer architecture course lectures and materials are freely available on YouTube, and his research group makes software artifacts freely available online. For more information, please see his webpage at http://people.inf.ethz.ch/omutlu/.

978-1-5386-5683-9/18/$31.00 ©2018 IEEE

REFERENCES

[1] U. Kang et al., Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling, in: The Memory Forum, 2014.
[2] O. Mutlu, Memory Scaling: A Systems Architecture Perspective, IMW (2013).
[3] O.Mutlu,L.Subramanian, Research Problems and Opportunities in Memory Systems, SUPERFRI (2014).
[4] S.A.McKee, Reflections on the Memory Wall, in: CF, 2004.
[5] M. V. Wilkes, The Memory Gap and the Future of High Performance Memories, CAN (2001).
[6] Y.Kim et al., Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, in: ISCA, 2014.
[7] Y. Kim et al., A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM, in: ISCA, 2012.
[8] J. Liu et al., RAIDR: Retention-Aware Intelligent DRAM Refresh, in: ISCA, 2012.
[9] O. Mutlu, The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser, in: DATE, 2017.
[10] D. Lee et al., Decoupled Direct Memory Access: Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM, in: PACT, 2015.
[11] B. C. Lee et al., Architecting Phase Change Memory as a Scalable DRAM Alternative, in: ISCA, 2009.
[12] H.Yoon et al., Row Buffer Locality Aware Caching Policies for Hybrid Memories, in: ICCD, 2012.
[13] H. Yoon et al. Efficient Data Mapping and Buffering Techniques for Multi-level Cell Phase-Change Memories, ACM TACO (2014).
[14] K. Lim et al., Disaggregated Memory for Expansion and Sharing in Blade Servers, in: ISCA, 2009.
[15] W.A.Wulf et al., Hitting the Memory Wall: Implications of the Obvious, CAN (1995).
[16] K. K. Chang et al., Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization, in: SIGMETRICS, 2016.
[17] D.Lee et al., Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture, in: HPCA, 2013.

[18] D. Lee et al., Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case, in: HPCA, 2015.
[19] K. K. Chang et al., Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms, in: SIGMETRICS, 2017.
[20] D. Lee et al., Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms, in: SIGMETRICS, 2017.
[21] Y. Luo et al., Characterizing Application Memory Error Vulnerability to Optimize Datacenter Cost via Heterogeneous-Reliability Memory, in: DSN, 2014.
[22] H. Hassan et al., SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies, in: HPCA, 2017.
[23] H. Hassan et al., Reducing DRAM Latency by Exploiting Row Access Locality, in: HPCA, 2016.
[24] V. Seshadri et al., RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization, in: MICRO, 2013.
[25] V. Seshadri et al., Fast Bulk Bitwise AND and OR in DRAM, CAL (2015).
[26] K. K. Chang et al., Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM, in: HPCA, 2016.
[27] V. Seshadri et al., Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM, arXiv:1611.09988 [cs:AR] (2016).
[28] V. Seshadri et al., Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology, in: MICRO, 2017.
[29] J. Ahn et al., A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing, in: ISCA, 2015.
[30] A. Boroumand et al., Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks, in: ASPLOS, 2018.
[31] K. Hsieh et al., Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems, in: ISCA, 2016.
[32] A. Pattnaik et al., Scheduling Techniques for GPU Architectures with Processing-in-Memory Capabilities, in: PACT, 2016.
[33] J. Ahn et al., PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture, in: ISCA, 2015.
[34] A. Boroumand et al., LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory, CAL (2016).
[35] K. Hsieh et al., Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation, in: ICCD, 2016.
[36] Z. Liu et al., Concurrent Data Structures for Near-Memory Computing, in: SPAA, 2017.