Retrospective: RAIDR: Retention-Aware Intelligent DRAM Refresh

Onur Mutlu
ETH Zürich

***Abstract*—Dynamic Random Access Memory (DRAM) is the prevalent memory technology used to build main memory systems of almost all computers. A fundamental shortcoming of DRAM is the need to refresh memory cells to keep stored data intact. DRAM refresh consumes energy and degrades performance. It is also a technology scaling challenge as its negative effects become worse as DRAM cell size reduces and DRAM chip capacity increases.**

Our ISCA 2012 paper, RAIDR [1], examines the DRAM refresh problem from a modern computing systems perspective, demonstrating its projected impact on systems with higher-capacity DRAM chips expected to be manufactured in the future. It proposes and evaluates a simple and low-cost solution that greatly reduces the performance & energy overheads of refresh by exploiting variation in data retention times across DRAM rows. The key idea is to group the DRAM rows into bins in terms of their minimum data retention times, store the bins in low-cost Bloom filters, and refresh rows in different bins at different rates. Evaluations in our paper (and later works) show that the idea greatly improves performance & energy efficiency and its benefits increase with DRAM chip capacity. The paper embodies an approach we have termed *system-DRAM co-design*.

This short retrospective provides a brief analysis of our RAIDR paper and its impact. We briefly describe the mindset and circumstances that led to our focus on the DRAM refresh problem and RAIDR's development, discuss later works that provided improved analyses and solutions, and make some educated guesses on what the future may bring on the DRAM refresh problem (and more generally in DRAM technology scaling).

I. Background, Approach & Mindset

At the time we began our focus on solving the DRAM refresh (i.e., data retention) challenge in late 2010, my research group, SAFARI, had already been working on memory controllers and memory technology scaling issues, motivated by many challenges memory systems, in particular the DRAM technology [2], have been facing (as described in, e.g., [3–5]). Our intense work on memory systems started during my tenure at Microsoft Research from 2006 and continued at CMU from 2009. For example, we had developed better memory schedulers for multi-core processors (e.g., [6–10]), developed platforms to perform voltage and frequency scaling to save DRAM energy (e.g., [11]) and architected emerging memory technologies to replace or augment DRAM (e.g., [12–14]). We were quite excited about the prospect of much more capable memory controllers in enabling better memory systems. As such, we were pursuing new memory-controller and system-level techniques to 1) overcome the challenging device- and circuit-level scaling issues of memory technologies and 2) better exploit underlying characteristics of memory technology; an approach we termed *system-DRAM co-design* [4, 5].

RAIDR is a product of this approach. Our focus on data retention issues and other low-level issues in DRAM especially increased via discussions with the Samsung DRAM Design Team, who visited us in April 2011 and encouraged the development of our system-level solutions to DRAM issues, enabling strong support both technically and funding-wise. In fact, much of our ensuing research in DRAM was supported by generous gift funding by and technical discussions with Samsung based on a proposal entitled *"New ideas to enhance DRAM scaling: Scaling-aware controller design and co-design of DRAM and controllers"* (Intel provided similar gift funding and technical discussions).

II. Contributions and Impact of RAIDR

RAIDR is the first work to propose a low-cost memory controller technique that reduces refresh operations by exploiting variation in data retention times across DRAM rows. Its appeal comes from its simplicity and low cost, enabled by the careful use of Bloom filters [15]. Exploiting the DRAM data retention time distribution [16], RAIDR can eliminate a very large fraction (e.g., $\sim$75% or more) of refresh operations with very small hardware cost at the memory controller.

Apart from the new technique it introduced, we believe the RAIDR paper made two other major contributions that have enabled a large number of future works and new ideas. First, it provided an empirical scaling analysis that clearly demonstrated the importance of the DRAM refresh problem in modern systems: if nothing is done about it, DRAM refresh would waste almost half of the throughput and half of the energy of a high-capacity 64-Gb DRAM chip! This analytical prediction encouraged more works in the topic area. Second, it demonstrated a methodical way of exploiting cell-level heterogeneous data retention times at the system (e.g., memory controller) level: if data retention times of DRAM rows are accurately known, the system can use them to optimize DRAM refresh and get rid of most refresh operations. This demonstration enabled other works to develop 1) methods for accurately determining DRAM data retention times and 2) other system-level approaches to optimize DRAM behavior using data retention time information.

III. Building on RAIDR and Making It Work

We believe RAIDR enabled a refreshing approach to DRAM refresh. Its largest contribution could be the works it has inspired that rigorously examined the questions of 1) how to perform accurate DRAM data retention time profiling, 2) how to overcome potential hurdles that stand in the way of obtaining accurate minimum data retention times, 3) how to reliably get rid of unnecessary refresh operations.

We wanted to make RAIDR work in a real system setting. To this end, collaboratively with Intel, we developed an FPGA-based flexible DRAM testing infrastructure [17] that enabled us to rigorously test data retention times of cells in real DDR3 DRAM chips. Using this infrastructure, later open sourced as SoftMC [18, 19] and DRAM Bender [20, 21], we experimentally examined practical issues that affect the accuracy (and performance) of DRAM data retention time profiling. We analyzed two major issues that make such profiling very challenging: 1) data pattern dependence (DPD) of retention times [17, 22], and 2) the variable retention time (VRT) phenomenon [17, 23, 24]. Our follow-up work, which appeared at ISCA 2013 [17], provides a detailed experimental analysis of these challenges in cutting-edge DRAM chips, demonstrating that ideas like RAIDR that depend on accurate identification of retention times are not easy to exploit in practice. Later works (e.g., [25–32]) developed new methods for making RAIDR-like techniques more practical by tackling especially the DPD and VRT problems and enhancing retention time profiling methods to work in the presence of DPD and VRT, usually by exploiting ECC techniques that have since become mainstream in DRAM chips (see [31–33]) to tolerate VRT [34].

The development of our flexible FPGA-based DRAM testing infrastructure also enabled experimental DRAM research in directions that are completely different from retention time profiling and refresh. These include studies that provided valuable experimental data on various DRAM characteristics, including RowHammer [20, 35–44], latency [45–48], voltage-latency-reliability relationship [49], power consumption and modeling [50]. Using this infrastructure, later research also demonstrated the ability of real off-the-shelf DRAM chips to perform data copy/initialization and bulk bitwise operations [51–55], implement physical unclonable functions [56], and generate true random numbers [57, 58]. We believe the investment we made to try to make RAIDR work using a real FPGA-based infrastructure helped us and the broader research community uncover many interesting characteristics of DRAM chips and propose new ideas to make DRAM-based systems more secure, reliable, efficient, and high performance.

Other later works provided refined models of DRAM refresh's impact on system performance (e.g., [59, 60]) and developed new

methods to reduce DRAM refresh's negative impact on performance & energy (e.g., [41, 59–67]). Our HPCA 2014 paper [59] developed a more refined projection of the effect of DRAM refresh as technology scales. AVATAR in DSN 2015 [26] and REAPER in ISCA 2017 [30] enabled more practical ways of exploiting heterogeneous retention times in the presence of VRT. Our recent work [66] shows that with a more flexible DRAM interface that gives some autonomy to DRAM chips, RAIDR can be more efficiently implemented inside the DRAM chip.

IV. Summary and Future Outlook

RAIDR is a nice example of how enthusiastic support from industry can foster new ideas that can open up many new analyses and other ideas. We were inspired by our deep technical discussions with especially Samsung and Intel, along with prior works that described DRAM technology scaling challenges (e.g., [3]) and that developed promising solutions (e.g., [68, 69]). Engineers from Samsung and Intel later wrote an insightful paper [34] on DRAM scaling challenges, which described refresh as a key problem and advocated a controller-DRAM co-design approach as we had been advocating [1, 4]. RAIDR was also a nice example of how teaching & research smoothly feed each other: much of the research was done as part of a group project in the Parallel Computer Architecture class I taught at CMU in Fall 2011.

Looking forward, DRAM technology scaling is getting worse and data retention will continue to be an important issue [34, 70]. The negative effects of DRAM refresh will be (and are being) exacerbated by other technology scaling issues like RowHammer [35] that require even more refreshes as a solution [41, 44, 71]. We believe there are a lot more new ideas and techniques to develop to minimize the impact of refresh on computing systems.

References

[1] J. Liu *et al.*, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.
[2] R. H. Dennard, "Field-effect Transistor Memory," 1968, US Patent 3,387,286.
[3] J. A. Mandelman *et al.*, "Challenges and Future Directions for the Scaling of Dynamic Random-Access Memory (DRAM)," *IBM JRD*, 2002.
[4] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," *IMW*, 2013.
[5] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems," *SUPERFRI*, 2014.
[6] O. Mutlu and T. Moscibroda, "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors," in *MICRO*, 2007.
[7] O. Mutlu *et al.*, "Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems," in *ISCA*, 2008.
[8] Y. Kim *et al.*, "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior," in *MICRO*, 2010.
[9] Y. Kim *et al.*, "ATLAS: A Scalable and High-performance Scheduling Algorithm for Multiple Memory Controllers," in *HPCA*, 2010.
[10] S. Muralidhara, "Reducing Memory Interference in Multicore Systems via Application-aware Memory Channel Partitioning," in *MICRO*, 2011.
[11] H. David *et al.*, "Memory power management via dynamic voltage/frequency scaling," in *ICAC*, 2011.
[12] B. C. Lee *et al.*, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *ISCA*, 2009.
[13] H. Yoon *et al.*, "Row Buffer Locality Aware Caching Policies for Hybrid Memories," in *ICCD*, 2012.
[14] J. Meza *et al.*, "Enabling Efficient and Scalable Hybrid Memories using Fine-granularity DRAM Cache Management," *CAL*, 2012.
[15] B. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *CACM*, 1970.
[16] K. Kim and J. Lee, "A new investigation of data retention time in truly nanoscaled DRAMs," *IEEE EDL*, vol. 30, no. 8, 2009.
[17] J. Liu *et al.*, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," in *ISCA*, 2013.
[18] H. Hassan *et al.*, "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.
[19] "SoftMC Repository," https://github.com/CMU-SAFARI/SoftMC/.
[20] A. Olgun *et al.*, "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," *TCAD*, 2023.
[21] "DRAM Bender," https://github.com/CMU-SAFARI/DRAM-Bender.
[22] Y. Nakagome *et al.*, "The impact of data-line interference noise on DRAM scaling," *JSSC*, 1988.
[23] D. Yaney *et al.*, "A Meta-stable Leakage Phenomenon in DRAM Charge Storage - Variable Hold Time," in *IEDM*, 1987.
[24] P. J. Restle *et al.*, "DRAM Variable Retention Time," in *IEDM*, 1992.
[25] S. Khan, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS*, 2014.
[26] M. K. Qureshi *et al.*, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems," in *DSN*, 2015.
[27] S. Khan *et al.*, "PARBOR: An Efficient System-Level Technique to Detect Data Dependent Failures in DRAM," in *DSN*, 2016.
[28] S. Khan *et al.*, "A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM," *CAL*, 2016.
[29] S. Khan *et al.*, "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content," in *MICRO*, 2017.
[30] M. Patel, "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," in *ISCA*, 2017.
[31] M. Patel *et al.*, "Bit-Exact ECC Recovery (BEER): Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics," in *MICRO*, 2020.
[32] M. Patel *et al.*, "HARP: Practically and Effectively Identifying Uncorrectable Errors in Main Memory Chips That Use On-Die ECC," in *MICRO*, 2021.
[33] M. Patel *et al.*, "Understanding and Modeling On-die Error Correction in Modern DRAM: An Experimental Study using Real Devices," in *DSN*, 2019.
[34] U. Kang *et al.*, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum*, 2014.
[35] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.
[36] J. S. Kim *et al.*, "Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques," in *ISCA*, 2020.
[37] L. Orosa, "A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses," in *MICRO*, 2021.
[38] M. Farmani *et al.*, "RHAT: Efficient RowHammer-Aware Test for Modern DRAM Modules," in *ETS*, 2021.
[39] P. Frigo *et al.*, "TRRespass: Exploiting the Many Sides of Target Row Refresh," in *S&P*, 2020.
[40] H. Hassan *et al.*, "Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications," in *MICRO*, 2021.
[41] A. G. Yağlikci *et al.*, "HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips," in *MICRO*, 2022.
[42] A. G. Yağlıkcı *et al.*, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices," in *DSN*, 2022.
[43] A. Olgun *et al.*, "An Experimental Analysis of RowHammer in HBM2 DRAM Chips," in *DSN Disrupt*, 2023.
[44] H. Luo *et al.*, "RowPress: Amplifying Read Disturbance in Modern DRAM Chips," in *ISCA*, 2023.
[45] D. Lee *et al.*, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," in *HPCA*, 2015.
[46] K. K. Chang *et al.*, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization," in *SIGMETRICS*, 2016.
[47] D. Lee *et al.*, "Design-induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms," *POMACS*, 2017.
[48] J. Kim *et al.*, "Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines," in *ICCD*, 2018.
[49] K. Chang *et al.*, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *SIGMETRICS*, 2017.
[50] S. Ghose *et al.*, "What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study," in *SIGMETRICS*, 2018.
[51] V. Seshadri *et al.*, "Fast Bulk Bitwise AND and OR in DRAM," *CAL*, 2015.
[52] V. Seshadri *et al.*, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.
[53] A. Olgun *et al.*, "PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM"," *TACO*, 2023.
[54] F. Gao *et al.*, "ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs," in *MICRO*, 2019.
[55] F. Gao *et al.*, "FracDRAM: Fractional Values in Off-the-Shelf DRAM," in *MICRO*, 2022.
[56] J. S. Kim *et al.*, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices," in *HPCA*, 2018.
[57] J. Kim, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput," in *HPCA*, 2019.
[58] A. Olgun *et al.*, "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAMs," in *ISCA*, 2021.
[59] K. K. Chang *et al.*, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," in *HPCA*, 2014.
[60] J. Mukundan *et al.*, "Understanding and Mitigating Refresh Overheads in High-Density DDR4 DRAM Systems," in *ISCA*, 2013.
[61] C.-H. Lin *et al.*, "SECRET: Selective Error Correction for Refresh Energy Reduction in DRAMs," in *ICCD*, 2012.
[62] P. J. Nair *et al.*, "ArchShield: Architectural Framework for Assisting DRAM Scaling by Tolerating High Error Rates," in *ISCA*, 2013.
[63] P. Nair *et al.*, "A Case for Refresh Pausing in DRAM Memory Systems," in *HPCA*, 2013.
[64] T. Zhang *et al.*, "CREAM: A Concurrent-Refresh-Aware DRAM Memory Architecture," in *HPCA*, 2014.
[65] H. Hassan *et al.*, "CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability," in *ISCA*, 2019.
[66] H. Hassan *et al.*, "A Case for Self-Managing DRAM Chips: Improving Performance, Efficiency, Reliability, and Security via Autonomous in-DRAM Maintenance Operations," arXiv:2207.13358, 2022.
[67] A. Das *et al.*, "VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency," in *DAC*, 2018.
[68] R. Venkatesan *et al.*, "Retention-Aware Placement in DRAM (RAPID): Software Methods for Quasi-Non-Volatile DRAM," in *HPCA*, 2006.
[69] S. Liu *et al.*, "Flikker: Saving DRAM Refresh-Power through Critical Data Partitioning," in *ASPLOS*, 2011.
[70] W. Kim, "A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Security and Reliability Enhancement," in *ISSCC*, 2023.
[71] O. Mutlu *et al.*, "Fundamentally Understanding and Solving RowHammer," in *ASP-DAC*, 2023.