RAIDR: Retention-Aware Intelligent DRAM Refresh

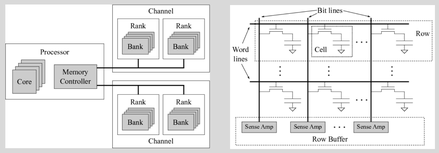
Ben Jaiyen, Jamie Liu & Richard Veras

Abstract

Dynamic random-access memory (DRAM) is the building block of main memory systems. A DRAM cell must be periodically refreshed in order to maintain data integrity. Refresh operations consume energy and degrade system performance. The negative effects of DRAM refresh increases as DRAM device capacity increases. Existing DRAM devices refresh all cells at a rate determined by the leakiest cell in the device. However, most DRAM cells can retain data for significantly longer. As a result, many of these refreshes are unnecessary.

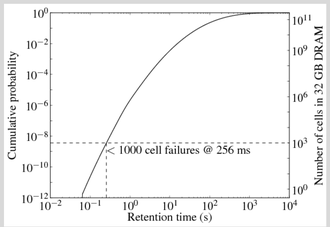
We propose RAIDR (Retention-Aware Intelligent DRAM Refresh), a low-cost mechanism that can identify and skip unnecessary refreshes using knowledge of cell retention times. Our key idea is to group DRAM rows into retention time bins and apply a different refresh rate to each bin, so that rows containing leaky cells are refreshed more frequently while most rows are refreshed less frequently. RAIDR uses Bloom filters to implement these retention time bins efficiently. RAIDR requires no modification to DRAM and minimal modification to the memory controller. In an 8-core system with 32~GB DRAM, RAIDR achieves an average DRAM operating power reduction of 16.1%, and an average system performance improvement of 6.9!% over existing systems, at a modest storage overhead of 1.25~KB in the memory controller. RAIDR's benefits are robust to variation in DRAM system configuration, and increase as core count and memory capacity increase.

DRAM Organization



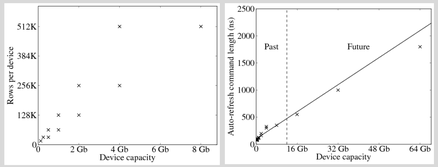
- DRAM cells store data in the form of charge on a capacitor.
- This charge leaks, causing the stored data to be lost.
- Data stored in DRAM must be periodically refreshed.
- DRAM refresh operations waste energy and degrade performance by delaying processor memory requests.

DRAM Cell Retention Time



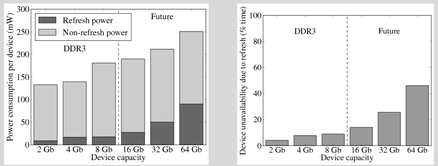
- In a system with 32 GB of DRAM, fewer than 1000 cells require a refresh interval shorter than 256 ms, which is four times the standard refresh rate of 64 ms.

Auto-Refresh Command Length



An auto-refresh operation occupies all banks on the rank simultaneously for a length of time t_{RFC}. **Left:** t_{RFC} scales linearly with DRAM device capacity. **Right:** This scaling is extrapolated for future devices.

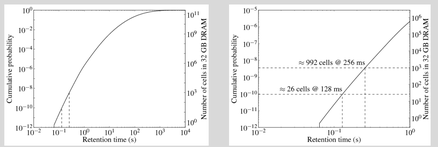
DRAM Refresh Impact on Future Devices



DRAM refresh operations waste and energy and degrade performance. This is expected to worsen over time.

Left: shows the proportion of DRAM power spent on refresh operations, while the figure on the **Right** shows the device throughput loss due to refresh operations, for current and projected future devices in the high-temperature range of dense server environments

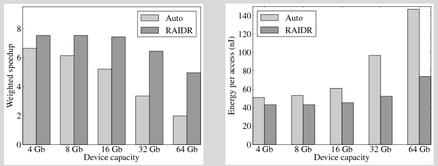
DRAM Retention Time Distribution



Left: Retention time distribution follows a log-normal distribution.

Right: The vast majority of cells can tolerate long refresh intervals.

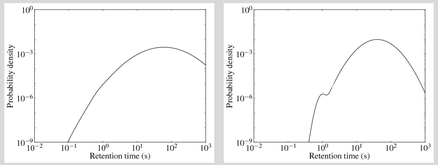
RAIDR Scales with Device Capacity



Left: As device capacity scales from 4 Gb to 64 Gb, the auto-refresh system's performance degrades by 70.3%, while RAIDR's performance degrades by 34.3%.

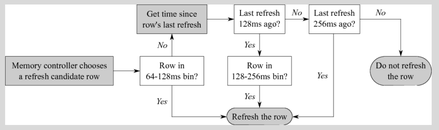
Right: As device capacity scales from 4 Gb to 64 Gb, the auto-refresh system's access energy increases by 187.6%, while RAIDR's access energy increases by 71.0%.

Trends in Retention Time Distribution



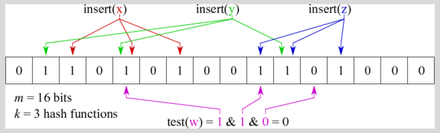
- DRAM scales to smaller technology nodes, both the normal and leaky parts of the retention time distribution will narrow.
- Since this means a decrease in the proportion of very weak cells in an array, we expect that RAIDR will not change in effectiveness.

RAIDR Overview



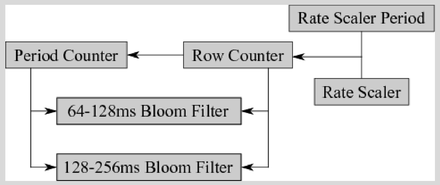
- We define a row's retention time as the minimum retention time across all cells in that row.
- A set of bins is added to the memory controller, each associated with a range of row retention times.
- Each bin contains all of the rows whose retention time falls into that bin's range. The shortest retention time covered by a given bin is the bin's refresh interval.
- The shortest retention time that is not covered by any bins is the new default refresh interval .

Bloom Filters



- A Bloom filter is a structure representing a set that can be implemented efficiently in hardware.

RAIDR in Detail



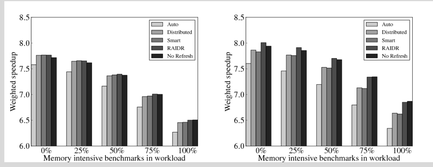
- RAIDR requires three counters, bit arrays to store the Bloom filters, hash functions to index the Bloom filters, and miscellaneous control logic.
- The longest counter, the row counter, is limited in length to the longest row address supported by the memory controller, which in current systems is on the order of 32 bits.
- The majority of RAIDR's hardware overhead will be in the Bloom filters.

System Configuration

Component	Specification
Processor	8-core, 4GHz, 3-wide issue, 128-entry instruction window, 16 MSHRs per core
Per-core cache	512 KB, 16-way, 64 B cache line size
Memory Controller	FR-FCFS scheduling, line-interleaved mapping, open-page policy
DRAM Organization	32 GB, 2 channels, 4 ranks/channel, 8 banks/rank, 64K rows/bank, 8 KB rows
DRAM device	64x Micron MT41J512M8RA-15E (DDR3-1333)

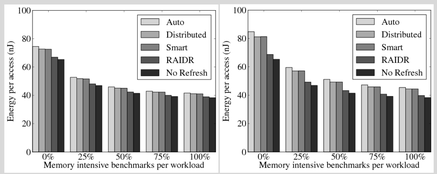
Retention Range	Rows n	Bloom filter size m	Numer of hashes k	False positive probability
64 ms – 128 ms	28	256 B	10	$1.16 * 10^{-9}$
128 ms – 256 ms	978	1 KB	6	0.0179

RAIDR Performance



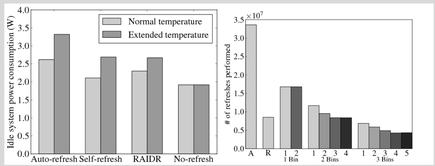
- RAIDR consistently provides significant performance gains in both the normal and extended temperature ranges, averaging a 3.2%/6.9% improvement over auto-refresh (normal temperature/extended temperature).
- This performance improvement increases as memory intensity increases, to an average improvement of 3.7%/8.0% over auto-refresh at 100% memory-intensive benchmarks.

RAIDR Power Consumption



- All studied refresh mechanisms result in a net energy per access decrease because the improvements in performance allow the memory to enter a low-power mode more frequently.
- RAIDR decreases energy per access by 8.4%/16.1% on average, within 2.2%/4.6% of the no-refresh ideal.

RAIDR Effectiveness



- Although using RAIDR during long idle periods results in the lowest DRAM power usage in the extended temperature range (a 19.6% improvement over idle auto-refresh).
- The self-refresh mechanism has lower power consumption in the normal temperature range.

References

[1] Inflant Corp., "Reducing server power consumption by 20% with pulsed air jet cooling," White paper, 2009.
[2] Intel Corporation, "Intel 64 and IA-32 Architectures Software Developer's Manual," 2010.
[3] T. Ohsawa, K. Kai, and K. Murakami, "Optimizing the DRAM refresh count for merged DRAM/logic LSIs," in Proceedings of the 1998 International Symposium on Low Power Electronics and Design, 1998.
[4] J. Kim and M. C. Papaefthymiou, "Dynamic memory design for low data-retention power," in Proceedings of the 10th International Workshop on Integrated Circuit Design, Power and Timing Modeling, Optimization and Simulation, 2000.
[5] J. Kim and M. C. Papaefthymiou, "Block-based multiperiod dynamic memory design for low data-retention power," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, no. 6, 2003.
[6] Micron Technology, Inc., "Calculating memory system power for DDR3," Technical note, 2007. Pierce, and N. Muzahid, "Retention Time Tolerance Bloom Filter," in ISCA-38, 2011.
[7] T. Kobayashi, C. J. Druckes, S. Morioka, and Z. Wu, "Fault-tolerant refresh power reduction of DRAMs for quasi-nonvolatile data retention," in Proceedings of the 14th International Symposium on Defect and Fault-Tolerance in VLSI Systems, 1999.
[8] J. Mukundan, H. Hunter, K.-h. Kim, J. Stuecheli, and J. F. Martínez, "Understanding and mitigating refresh overheads in high-density DDR4 DRAM systems," in ISCA '13, 2013.
[9] C. Wilkerson, A. R. Alameldeen, Z. Chishti, D. Gemaehlen, and S.-L. Lu, "Reducing cache power with low-cost, multi-bit error-correcting codes," in ISCA '37, 2010.
[10] R. K. Venkatesan, S. Herr, and E. Rotenberg, "Retention-aware placement in DRAM (RAPID): Software methods for quasi-non-volatile DRAM," in HPCA '12, 2006.
[11] C. Isen and L. K. John, "ESKIMO: Energy savings using semantic knowledge of inconsequential memory occupancy for DRAM subsystem," in MICRO-42, 2009.
[12] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: Saving DRAM refresh-power through critical data partitioning," in ASPLOS-16, 2011.
[13] J. Stuecheli, D. Kaseridis, H. C. Hunter, and L. K. John, "Elastic refresh: Techniques to mitigate refresh penalties in high density memory," in MICRO-43, 2010.
[14] K. Kim and J. Lee, "A new investigation of data retention time in truly nanoscaled DRAMs," IEEE Electron Device Letters, vol. 30, no. 8, 2009.
[15] B. Keeth, R. J. Baker, B. Johnson, and F. Lin, DRAM Circuit Design: Fundamental and High-Speed Topics. IEEE Series on Microelectronic Systems, Wiley-Interscience, 2008.
[16] JEDEC Solid State Technology Association, "DDR SDRAM Specification," 2008.
[17] JEDEC Solid State Technology Association, "DDR2 SDRAM Specification," 2009.
[18] JEDEC Solid State Technology Association, "DDR3 SDRAM Specification," 2010.
[19] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G.-H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb, "Die stacking (3d) microarchitecture," in MICRO-39, 2006.
[20] Micron Technology, Inc., "Various methods of DRAM refresh," Technical note, 1999.
[21] Micron Technology, Inc., "Calculating memory system power for DDR3," Technical note, 2007.
[22] T. Hamamoto, S. Sugiura, and S. Sawada, "On the retention time distribution of dynamic random access memory (DRAM)," IEEE Transactions on Electron Devices, vol. 45, no. 6, 1998.
[23] K. U. N. Schroeder, F. Schrodel, R. Thewes, and D. Schmitt-Landsiedel, "DRAM yield analysis and optimization by a statistical design approach," IEEE Transactions on Circuits and Systems I: Regular Papers, 2011. Preprint.
[24] Y. Nakagome, M. Aoki, S. Ikenaga, M. Horiguchi, S. Kimura, Y. Kawamoto, and K. Itoh, "The impact of data-line interference noise on DRAM scaling," IEEE Journal of Solid-State Circuits, vol. 23, no. 5, 1988.
[25] J. L. Carlson, Redundancy Analysis for Memory Arrays. PhD thesis, Rutgers, 1995.
[26] M. Ahmadi and G. Wang, "5-stage pipelined Bloom filter for packet classification," in Proceedings of the 16th International Conference on Computational Science and Engineering, 2009.
[27] M. J. Lyons and D. Brooks, "The design of a Bloom filter hardware accelerator for ultra low power systems," in Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design, 2009.
[28] D. E. Knuth, The Art of Computer Programming, vol. 3 Addison-Wesley, 2nd ed., 1998.
[29] L. L. Larmore and T. Hirschberg, "A fast algorithm for optimal length-limited Huffman codes," in Proceedings of the 9th Annual ACM Symposium on Theory of Computing, 1977.
[30] L. Carter, R. Floyd, J. Gill, G. Markowsky, and M. Wegman, "Exact and approximate membership testers," in STOC '78, 1978.
[31] M. Dietzfelbinger, T. Hagerup, J. Katajainen, and M. Penttonen, "A reliable randomized algorithm for the closest-pair problem," Journal of Algorithms, vol. 25, no. 1, 1997.
[32] G. Marsaglia, "Xorshift RNGs," Journal of Statistical Software, vol. 8, no. 14, 2003.
[33] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," in Allerton, 2002.
[34] P. G. Emma, W. R. Reohr, and L. A. Wang, "Restore tracking system for DRAM," U.S. patent, 2002. Patent number 6389505.
[35] S. P. Song, "Method and system for selective DRAM refresh to reduce power consumption," U.S. patent, 2000. Patent number 6094705.
[36] K. Patel, E. Macii, M. Poncino, and L. Benini, "Energy-efficient value based selective refresh for embedded DRAMs," Journal of Low Power Electronics, vol. 2, no. 1, 2006.
[37] J. Stuecheli, D. Kaseridis, H. C. Hunter, and L. K. John, "Elastic refresh: Techniques to mitigate refresh penalties in high density memory," in MICRO-43, 2010.
[38] B. Jacob, S. W. Ng, and D. T. Harper, Memory Systems: Cache, DRAM, Disk. "Memory system simulator," IEEE Computer Architecture Letters, vol. 10, no. 1, 2011.
[39] Standard Performance Evaluation Corporation, "SPEC CPU2006," 2006.
[40] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. Janapa, and K. K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in PLDI '05, 2005.
[41] Micron Technology, Inc., "4Gb: x4, x8, x16 DDR3 SDRAM," Data sheet, 2011.
[42] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu, "Mini-rank: Adaptive DRAM architecture for improving memory power efficiency," in Proceedings of the 41st IEEE/ACM International Symposium on Microarchitecture, 2008.
[43] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," in ISCA '27, 2000.
[44] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "CACTI 5.1," Tech. Rep. HPL-2008-20, HP Laboratories, 2008.
[45] T. Moscibroda and O. Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," in USENIX Security, 2007.
[46] Y. Kim, D. Han, O. Mutlu, and M. Harchol-Balter, "ATLAS: A scalable and high-performance scheduling algorithm for multiple memory controllers," in ISCA-16, 2010.