1 Review “Self-Optimizing Memory Controllers: A Reinforcement Learning Approach” + Retrospecitve (1)

Summary

Paper

The paper discusses off-chip DRAM bandwidth and proposes a reinforcement learning (RL) approach to optimize DRAM scheduling. It simulates the proposed systems and discusses the results. The paper does this in the following steps:

- The paper establishes that previous DRAM scheduling was based on ad hoc heuristics such as first-ready first-come first-served (FR-FCFS). These are chosen as the best working solutions on an average of performance cases.
- The paper proposes an online RL approach which works by integrating an RL algorithm into the DRAM-controller which takes a set of inputs from its environment and based on their state decides which action to schedule.
- The RL approach has two main changes to the previous approach. It optimises based on a set of parameters given by the system and is not an ad hoc chosen heuristic. While this does not ensure optimal performance it turns out to be a better approach and can be tuned by modifying the learning behaviour of the RL algorithm. Further the RL approach is an online method which means that it can adapt to changes in the needs of the running program.
- The paper proposes a theoretical hardware implementation and evaluates the hardware overhead as the logic required to compute the state attributes, the logic required to estimate and update the Q-values in the RL model, and the SRAM to store the Q-values.
- The evaluation shows that the RL approach outperforms FR-FCFS on all single core systems and on all but one multi core system. The paper shows that even an offline RL approach outperforms FR-FCFS. Further the paper shows that these performance gains are due to the RL method and not due to the additional information granted to the RL-algorithm by creating an FR-FCFS version that has the same level of awareness of the system.

Retrospective

The retrospective provides context for the paper mainly as to the process of coming up with and exploring the RL approach. It also outlines the significance of the paper and its impact on future research and systems.

Strengths

The paper correctly assessed the need for innovation in DRAM scheduling and proposes a strong and performant solution for the issue. The RL solution and giving the DRAM controller a higher level of awareness of its environment is a smart approach to increase the efficiency. The results of the paper and its level of impact stand for themselves in showing the relevance of the research presented.

Weaknesses

While the paper provides an overview of the physical overhead it does not provide a full picture meaning that the proposed solution was fairly far away from an actual implementation. The paper only proposes a very basic approach to RL to optimise the performance of the DRAM controller but a more sophisticated approach could yield far better results.

My POV

The paper clearly was an important milestone in the development of modern DRAM controllers. I do not see a specific criticism to the paper but it is clear that the method due to its novelty has not reached the level of sophistication required for an actual implementation in a real system. This is the only direct issue I have with the paper and it is one that was alleviated by follow-up research.

Takeaways

- While ad hoc heuristics can give a good baseline performance they are an indicator for a possible system inefficiency.

1 Review "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors" + Retrospecitve (2)

Summary

Paper

The paper discusses the concept of user induced bit-flips in DRAM memory. It investigates its causes as well as its security implications. The paper demonstrates the prevalence of these bit-flips on commercial DRAM chips and proposes a set of possible mitigation strategies. It does this in the following steps:

- The paper introduces the concept of RowHammer. The basic idea is that the state of a DRAM cell can be influenced by repeatedly loading adjacent DRAM-rows. This can be executed by user programs meaning that an unprivileged user can corrupt the memory of any system that uses vulnerable DRAM.
- The paper tested RowHammer on a broad set of DRAM models from three unspecified manufacturers ranging from the year 2010 to the year 2014. The paper's results show that the bit errors only occur in more modern chips the manufacturing times of which align with process upgrades by the respective manufacturers. This is explained to be a consequence of more modern smaller technology being more vulnerable due to the lower capacitance of the cells and denser packing on the chip.
- The paper also demonstrates that RowHammer mainly causes bit-flips from physical 1 to physical 0 meaning a decrease in voltage. It also shows that counterintuitively there is no correlation between "weak cells" and RowHammer victim cells. The paper also shows that RowHammer is "strong enough" to circumvent error correction as commonly implemented.
- The paper proposes seven possible mitigation strategies to RowHammer:
  - Make better chips
  - Correct errors
  - Refresh all rows frequently
  - Retire cells (manufacturer)
  - Retire cells (end-user)
  - Identify "hot" rows and refresh neighbours.
  - PARA (probabilistic adjacent row activation)
- The paper outlines that the first six all come at significant cost and proposes PARA as the "best" solution. PARA is a methodology that randomly refreshes adjacent rows when a row is activated. The paper outlines that to implement this the manufacturers have to share their mapping functions of the DRAM as well as their remapping methodology. This is necessary to identify adjacent rows.

Retrospective

The retrospective gives context to the creation of the paper and outlines the cooperation between the researchers and the industry. It also discusses the widespread impact of RowHammer outlining that both manufacturers as well as other industry participants performed their own research and proposed solutions as well as methods to exploit RowHammer.

Strengths

The paper was the first to draw attention to a big gap in hardware security. It clearly demonstrates the issue and its prevalence on a wide variety of chips. It further shows that RowHammer is an issue only becoming more important over time due to smaller technology being more vulnerable to RowHammer attacks. The paper proposes a good set of strategies to deal with RowHammer and comments on their effectiveness and feasibility. Mainly its PARA proposal seems to have gained traction in later works as well as industry implementations.

Weaknesses

The main weakness of the paper is the proposed solution PARA. It uses a set probability p to decide whether a refresh is performed. But as there are no true random processes in a computer it is possible to reconstruct the coin-flip. If a malicious actor can deduce the coin-flip then he can completely circumnavigate the PARA security method by switching to another line whenever a refresh is imminent.

My POV

The RowHammer paper is a seminal work that has caused much discussion and change of thought in the industry. While its solutions might not be perfect it still draws attention to an important topic and exposes the vulnerability of computer systems. By highlighting this weakness it has become an important contribution to modern computer architecture.

Takeaways

- RowHammer attacks are both dangerous and hard to prevent. While they are still complicated to implement it is clear that the threat of RowHammer attacks does not decrease with ever smaller technology.

3 “RowPress: Amplifying Read Disturbance in Modern DRAM Chips” (3)

Summary

The paper discusses read disturbance effects in DRAM chips. The specific effects they discuss are bit flips caused by keeping a DRAM row open for a long time. The paper also discusses the characteristics of these read disturbance phenomenon and differentiates it to similar effects such as retention-time and RowHammer. The paper also discusses possible solutions to mitigate the effects of these read disturbance effects which are named RowPress. This is done in the following steps.

- The paper outlines the occurrence of read disturbance effects induced by running a RowHammer extended row open timing. They coin the term RowPress to describe an attack that causes bit flips by keeping an aggressor row which is physically adjacent to a victim row open for a long period of time.
- The paper shows that this effect is fundamentally different from both retention time and RowHammer for the following reasons:
  - In previous works it was shown that RowHammer and retention time do not affect the same DRAM cells, i.e. a cell with a weak retention time is not more or less likely to be a RowHammer victim. And a RowHammer victim cell is not more or less likely to suffer from short retention time. The paper shows that RowPress victims are also not correlated to RowHammer victims or weak cells. This means RowPress is indeed a separate phenomenon from RowHammer and retention time.
  - The paper provides data that shows that RowPress is dependent on environment temperature. In general a higher temperature decreases the minimal time an aggressor row needs to be turned on, in a RowPress attack interval, to cause a bit flip in the victim row. This is again different from RowHammer which showed no correlation with environment temperature.
- The paper therefore establishes that RowPress is a distinct read disturbance phenomenon which occurs in modern DRAM chips. It further shows that the occurrence of RowPress gets worse with newer technology, i.e. smaller technology nodes.
- The paper demonstrates that a simple user side C program can induce RowPress bit flips in a system and therefore concludes that RowPress is a significant vulnerability in modern systems. It also shows that current RowPress mitigation strategies do not prevent RowPress attacks.
- The paper proposes four strategies to mitigate the risks of RowPress attacks:
  1. Error Correcting Codes (ECC)
  2. Decoupling the Row Buffer from the Row
  3. Limiting the Maximum Row-Open Time
  4. Adapting Existing RowHammer Mitigations

  The paper elaborates that the first three have prohibitively large cost and therefore proposes to follow the fourth option. This is done by reducing the RowHammer threshold of the existing RowHammer defense to account for the required activations which would cause a RowPress bit flip.

Strengths

The paper clearly demonstrates the dangers and occurrence of RowPress and differentiates it from RowHammer and retention time effects. It clearly outlines the possibility of user-level RowPress attacks on a system as well as the environment and technology dependence of RowPress. The paper also finds a working and easily implementable mitigation strategy to the problem.

Weaknesses

The discussion of the mitigation strategies is to short and lacking nuance. The paper selects four possible solutions where the purpose of three of them is only to demonstrate their inefficiency in solving the problem.

My POV

In my opinion the paper provides an important perspective on the problem of RowPress. From the footnotes and related works it seems that the problem itself is already known to the industry but the paper provides an important in-depth analysis of the issue and proposes an efficient solution.

Takeaways

- RowPress induced read disturbance is an important issue that needs addressing in modern DRAM chips.
- The issue of RowPress (as well as RowHammer and retention time) gets worse with smaller technologies. Therefore it is important to address these issues in modern systems and keep looking for possible attack vectors of malicious actors in modern computer systems.

4 Review "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling" (17)

Summary

The paper outlines two key issues facing modern DRAM design.

First: Refresh is the process of recharging DRAM cells as they lose their charge due to leakage currents. Therefore each cell has a retention time (tRET) which states how long it can retain a value.

Second: Write recovery time (tWR), which is the time it takes to write to the DRAM. More specifically it is the time after the last write data burst to the precharge command issued by the same bank.

Third: Variable retention time (VRT), which is the difference of retention time of different cells.

The paper shows that the retention time is significantly worse for higher temperatures and for newer technology as the smaller scale of the new technology nodes increases the leakage current while also decreasing the capacity of the transistors.

The paper proposes the co-desing of the DRAM and its controller to enhance the scaling of the DRAM process. To this effect it outlines three approaches which could manage the outlined issues.

First: Sub-array level parallelism (SALP), the idea is to use the different page buffers to store information from the same bank in parallel.

Second: Temperature compensated tWR, the idea is to use different tWR for different temperatures. This works as for lower temperatures tWR needs to increase but for high temperatures it can be lowered.

Third: In-DRAM ECC with dummy data pre-fetching, for this on a read the DRAM controller fetches more data than is needed and uses this additional data to be able to perform ECC with less size overhead.

Strengths

The paper clearly identifies issues which need to be addressed in modern DRAM systems and also proposes solutions for them. While the solutions themselves are not perfect the paper still shows the importance and benefit of using co-designed controllers and DRAM. The idea of using SALP to alleviate the issue of tWR.

Weaknesses

Some of the solutions proposed are not fully thought through. The idea of using a variable tWR only works if there is a good profiling of the DRAM chip and the necessary tWR for each cell. Further while the use of ECC is promising it might not be sufficient for high workload DRAM chips which run at high temperatures.

My POV

There are clear advantages to co-designing DRAM and the controller. But the solutions proposed, while good starting points, still need improving. While the idea of lowering tWR depending on temperature seems like a nice idea it makes me worry about the case of variable tWR in the same sense as variable VRT. While I'm not an expert on the subject I see a clear need of further research.

Takeaways

- Co-designing DRAM and controllers can give significant improvements in performance to the point that it is a necessity for modern technology nodes.

5 Review “Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator” (18)

Summary

The paper introduces the DRAM simulator Ramulator 2.0. Ramulator 2.0 builds on the foundations of Ramulator (referred to as Ramulator 1.0). Ramulator 2.0 is a fast and easily modifiable DRAM simulator implemented in C++20. Ramulator 2.0 makes the following improvements on its predecessors.

First: Ramulator 2.0 is highly modular. Individual parts of the simulation are implemented as standalone modules that communicate over standardized interfaces. This has the major advantage that intrusive changes to the simulated system can be made easily by modifying or replacing the C++ class which implements the functionality.

Second: Ramulator 2.0 provides a simple human-readable way to handle DRAM specifications. This simplifies the process of changing and testing different specification permutations significantly.

Ramulator 2.0 has been verified on MICRO’s DDR4 verilog model to ensure correct functionality. Further the paper demonstrates the implementations and simulations of a wide array of RowHammer mitigation techniques to demonstrate the ease of modification of Ramulator 2.0.

Strengths

Ramulator 2.0 is a cycle accurate, fast, verified, and easily modifiable DRAM simulator. This is a significant feat that simplifies future research and verification significantly. The decision to open-source Ramulator 2.0 is also very commendable.

My POV

I see no concrete downsides to Ramulator 2.0. There might be bugs or other inconveniences but as I have not used the tool I cannot comment on such things. From the paper it seems like a fast, open-source, and user-friendly DRAM simulator.

Takeaways

- The capability to simulate DRAM behaviours with different configurations and the capability to freely adapt a system is very important. Ramulator 2.0 seems to provide this.

6 Review “Artifact Appendix: GPUHammer: Rowhammer Attacks on GPU Memories are Practical” (50)

Summary

This is not a paper in itself but an appendix to the paper “GPUHammer: Rowhammer Attacks on GPU Memories are Practical”, this appendix gives a concrete implementation of a Rowhammer attack on a GPU system. The text clearly outlines what prerequisites are needed and provides scripts to execute the Rowhammer attack on a GPU system. It also gives a nice breakdown of the workflow.

- **ConflictRow-Set Generation:** The text provides a script which generates the Row-Set and Conflict-Set for the banks that will be hammered. The authors estimate that this step takes roughly one day to execute.
- **Systematic Hammering Campaigns:** A script is provided which performs 24-sided hammering patterns on the entire memory. The authors estimate that this steps takes roughly one day.
- **Bit-flip Characterization:** A set of experiments to characterize the Rowhammer Threshold and the TRR Sampler Size. The authors estimate that this step takes roughly five hours.
- **ML Model Exploit:** A script is provided which implements a specific exploit to degrade the accuracy of ML models through Rowhammer bit-flips. This is done by force placing model weights in victim locations followed by hammering.

Strengths

The authors provide a working Rowhammer attack script and show how it can degrade the performance of a real world system (ML Model), on a GPU. The paper clearly demonstrates the timescales needed to produce an effective Rowhammer attack.

Weaknesses

The functioning of the exploit depends on a lot of knowledge of the system and the impact on the ML model are only visible if the weights are purposefully placed in victim locations.

My POV

The paper is clearly a proof of concept, which shows that real Rowhammer attacks on GPU systems work. While it does not provide a malicious script, much rather it provides a script that can only be realistically be run by the system administrator, it is still clear that a similar and more sophisticated script could be used by a malicious actor to achieve the same effect. This clearly outlines the danger of Rowhammer attacks and in our current context the risk posed to big AI data centers which depends in large part on GPUs or accelerators with similar designs as GPUs.

Takeaways

- Rowhammer attacks work not only on RAM but also on GPU memory (VRAM).
- Rowhammer attacks can be performed in relatively short time scales (days).
- Rowhammer attacks can effectively be used to influence real-world applications such as ML algorithms.

7 Review "An Experimental Characterization of Combined RowHammer and RowPress Read Disturbance in Modern DRAM Chips" (45)

Summary

The paper evaluates a method to combine RowHammer and RowPress read disturbance on modern DRAM chips. For this purpose the method was tested on a wide array of different DRAM chips from all three major DRAM manufacturers. The paper makes the following key observations:

- As the time the aggressor row is turned on initially increases, the combined RowHammer and RowPress pattern takes much less time to induce the first bitflip compared to both the conventional single- and double-sided RowPress patterns.
- As the time the aggressor row is turned on initially starts to increase, the combined pattern needs slightly more aggressor row activations to induce at least one bitflip than the conventional double-sided RowPress pattern.
- As the aggressor-on time continues to increase, the combined pattern takes a similar amount of time to induce the first bitflip as the conventional single-sided RowPress pattern.
- As the aggressor-on time increases, the directionality of bitflips caused by the combined RowHammer and RowPress pattern changes.
- The overlap between the bitflips from the combined pattern and conventional single-sided RowPress pattern increases as the aggressor-on time increases.
- The overlap between the bitflips from the combined pattern and conventional double-sided RowPress pattern first decreases as the aggressor-on time initially starts to increase, and then increases as the aggressor-on time continues to increase.

From these Observations the paper poses two hypotheses:

- As the aggressor-on time initially starts to increase, the read disturbance effect caused by RowPress from one of the two aggressor rows in the double-sided pattern is much more significant than the other.
- For large aggressor-on time values, the read disturbance effect from RowPress is dominant compared to RowHammer in the combined RowHammer and RowPress pattern.

The paper outlines two takeaways:

- Read disturbance bitflips can be induced in a smaller amount of time by combining RowPress and RowHammer compared to using solely RowPress or RowHammer.
- The combined RowHammer and RowPress pattern induces different bitflips compared to the conventional single- and double-sided RowPress patterns.

Strengths

The paper has a rigorous and broad minded approach. It successfully demonstrates that combined RowHammer and RowPress is more effective than only RowHammer or RowPress in isolation. The paper also provides a good analysis of the different appearing bitflip patterns, i.e. the change of bitflip direction with increasing aggressor-on time.

Weaknesses

The paper only provides a short introduction into how the combined algorithm works which only works well if one is already well versed in the subject of read disturbance.

My POV

The paper is a straight-forward analysis of a specific read disturbance scheme and its implementation on a set of DRAM chips. It clearly demonstrates the dangerous efficiency of combining RowHammer and RowPress to induce bitflips.

Takeaways

- Combining RowPress and RowHammer provides a scheme which is faster in causing bitflips than using only RowPress or only RowHammer.
- In the combined scheme the directionality of bitflips changes with increasing aggressor-on time.

8 Review “Revisiting DRAM Read Disturbance: Identifying Inconsistencies Between Experimental Characterization and Device-Level Studies” (40)

Summary

The paper studies the differences between experimental characterization and device-level studies of read disturbance effects. It does this by running a set of read disturbance attacks on 96 different DRAM chips covering all major DRAM manufacturers. The paper outlines a major characteristic for each of the two main read disturbance effects (RowHammer and RowPress), which stem from prior works covering device-level read disturbance mechanisms.

- **RowHammer:** Double-sided RowHammer should induce only “1” to “0” bitflips.
- **RowPress:** Single-sided RowPress should induce both “1” to “0” and “0” to “1” bitflips.

The paper makes four relevant observations:

- Double-sided Row Hammer induces both “0” to “1” and “1” to “0” bitflips.
- For Double-Sided RowHammer the HC_first values of “0” to “1” bitflips are significantly smaller than that of the “1” to “0” bitflips.
- Double-Sided RowHammer induces significantly more “1” to “0” bitflips than “0” to “1” bitflips at maximum aggressor row activation count.
- Single-sided RowPress overwhelmingly induces “1” to “0” bitflips.

Based on these observations the paper provides four takeaways:

- Double-sided RowHammer involves error mechanisms for inducing both “0” to “1” and “1” to “0” bitflips.
- For double-sided RowHammer, the observed error mechanism for “0” to “1” bitflips is stronger than “1” to “0” bitflips in the most vulnerable DRAM cells.
- For double-sided RowHammer, significantly more DRAM cells are vulnerable to the error mechanism for “1” to “0” bitflips than “0” to “1” bitflips when the aggressor rows are hammered enough times.
- For single-sided RowPress, the observed error mechanism of “1” to “0” bitflips is much stronger than that of “0” to “1” bitflips.

Based on this the paper concludes that there are three inconsistencies:

- The experimental characterization shows that double-sided RowHammer induces both “1” to “0” bitflips and “0” to “1” bitflips, which contradicts the first characteristic outlined at the beginning.
- The experimental characterization shows that double-sided RowHammer induces the initial “0” to “1” bitflips easier than the initial “1” to “0” bitflips, which contradicts the first characteristic outlined at the beginning.
- The experimental results show that even with a long aggressor row open time, a high aggressor row activation count , and using both the NWL and PWL as the aggressor rows, the overwhelming majority of single-sided RowPress bitflips are “1” to “0” bitflips, which contradicts the second characteristic outlined at the beginning.

The paper provides two hypotheses where these inconsistencies could arise:

- Existing research on the device-level error mechanisms of DRAM read disturbance are not comprehensive enough to cover all the major leakage mechanisms.
- The state-of-the-art true- and anti-cell reverse engineering technique based on DRAM cell retention failures is incorrect.

Strengths

The paper clearly demonstrates the inconsistencies in current research and provides solid data to back-up its claims that either device-level research is insufficient or reverse engineering is insufficient.

Weaknesses

The paper speculates about potential explanations of these inconsiatencies. Most of the proposed ideas are down to insufficient device-level research. The paper fails to realise that it could simply be a case where both the reverse engineering and device-level research is insufficient. In particular on the reverse engineering side, the manufacturers have a clear incentive to make this as hard as possible, so it might very well be that they adjusted their manufacturing processes such that the specific reverse engineering approach does not work.

My POV

The paper makes a strong case in pointing out the inconsistencies but only provides a very limited analysis of the reasons thereof. From my point of view, while there certainly is a need for more detailed research on the device-level side, even the best simulations at the moment are only approximations, there should also be a concerted effort at looking at the reverse engineering techniques as this is a constant arms race between reverse engineering and manufacturers.

Takeaways

- There are inconsistencies between device-level studies of read disturbance in DRAM and experimental results.
- Both additional device-level research is needed as well as a review of the state-of-the-art reverse engineering techniques.

9 Review “Panopticon: A Complete In-DRAM Rowhammer Mitigation” (54)

Summary

The paper first starts by outlining that previous Rowhammer mitigation techniques incur prohibitively high costs on the manufacturers which is why none of them have been adopted widely. The paper proposes Panopticon as a complete In-DRAM Rowhammer mitigation technique, i.e. it does not require changes to the system stack, especially the memory controller, it does not require the manufacturers to share proprietary information about victim cells, it does not require large amounts of SRAM or other types of additional memory, and it does not violate DDR4 protocols. The implementation uses some key ideas:

- Panopticon maintains a counter table in-DRAM where each counter corresponds to a DRAM row and increments each time the corresponding row is activated.
- Panopticon keeps a Rowhammer threshold bit instead of a Rowhammer threshold value, this removes the need to reset the counters as this is an expensive operation in DRAM.
- Panopticon keeps a service queue. When a threshold bit is flipped, i.e. the counter for a row is full, this row is enqueued in the service queue. Panopticon issues the ALERT signal to the memory controller to create space for the mitigation refreshes.
- Panopticon does not fully reset its counters when a regular refresh happens to a line but clears some of the counters lower bit values. The precise count depends on the implementation.
- Physically Panopticon comprises of a set of narrow DRAM mats which store the counters and a small piece of logic for each counter mat as well as a state machine in each bank that implements the service queue and the ALERT signaling.

Strengths

The paper provides a strong solution which is limited to DRAM only and does not violate DDR4 protocols or create the need to share proprietary information about the DRAM. Panopticon uses some very strong ideas to simplify its logic, such as using a threshold bit instead of a threshold value as well as the clearing of lower order bits to decrease the counters.

Weaknesses

Panopticon is implemented in DRAM. Depending on the implementation it itself could be susceptible to RowHammer or RowPress attacks which especially in the case of targeting the counters could enable denial of service (DoS) attacks. The proposed design of Panopticon leaves open space on the DRAM chip which means wasted space in an already tight environment. While the authors reason that this provides space for testing logic, etc. this rather seems like a detriment to the implementation and its viability.

My POV

While the paper makes a strong case for Panopticon, the two weakneses outlined above are significant. To mitigate the first a sufficient degree of separation in DRAM is needed between the counters and the rest of the DRAM but this in itself worsens the second issue which is the inefficient space usage. But it could be possible that a smart layout might solve these issues. The paper provides no results on the actual performance of the proposed system.

Takeaways

- While many Rowhammer mitigation techniques have been proposed, none have found widespread application. The reason for this is their prohibitive cost.
- Panopticon provides a solution to this issue and employs smart ideas to reduce the implementation cost of Rowhammer mitigation.
- Panopticon still has shortcomings and further research is clearly needed in the area of Rowhammer mitigation.