Novel STT-RAM Based Hybrid Cache for Intermittent Powered Processors in the IoT Devices

Presenter: Sam

October 5, 2020

1 Introduction

This paper proposed applying previously proposed STT-RAM/SRAM hybrid caches to a new use case: IoT devices. Using a hybrid cache in an IoT device requires frequent checkpointing due to their often intermittent power supply. This paper presents a caching algorithm

2 Architecture

The architecture proposed consists of entirely non-volatile memory except for the hybrid cache which is split between non-volatile STT-RAM and volatile SRAM. The cache simulated is 16kB in size with 64B blocks. Additionally, they run their simulations with a single core CPU with 480MHz frequency.

2.1 Hybrid Cache

Their cache is split into 4 segments, 2 STT-RAM and 2 SRAM, and is 4-way set associative. The cache is also bidirectional, which means data can move from SRAM to STT-RAM (when checkpointing) or STT-RAM to SRAM (if it becomes write heavy). The crux or their implementation was their pattern predictor, which is meant to dictate their cache placement and migration policies. Their implementation uses a Program counter (PC) and write program counter (W-PC) to track previous read and write accesses which are the basis of the prediction. Overall, the idea of the pattern predictor is valid. However, their presentation had some flaws. Below are a few key factors that their paper failed to detail:

1. They adapted their pattern predictor from paper that only has PC and not W-PC, but show no experimental results to back up the use of W-PC (Sam)

2. The paper does not address what happens if an incorrect pattern is predicted. Their paper seems to always assume that these predictors are always accurate (Iris)

3. It would have been better if paper included an evaluation of the accuracy of the pattern predictor since it was the basis of the paper (Semanti)

4. They do not address the overhead of the predictor. How long of a warm up is required for the predictor to work? What happens while the predictor is still warming up and not predicting accurately? (Jiwon)

5. Previous work applies the predictor to only a subset of the cache sets. Here, they seem to apply the predictor to the entire cache, which could be reasonable since this cache is small in comparison to previous work (which use it in a LLC), but they never say this explicitly or justify it in any way. (Jiwon)

Another key contribution of this paper was their checkpointing algorithm for moving blocks from SRAM to STT-RAM and STT-RAM to main memory. At a checkpoint, all dirty blocks in SRAM must be moved to STT-RAM. If blocks must be evicted from STT-RAM to accommodate these SRAM blocks, they are evicted in the following order:

1. Clean Blocks
2. Dead Dirty Blocks
3. Live Dirty Blocks

A question that was brought up in discuss asked if this eviction policy was costly given that it potentially requires multiple passes of the cache (if you need to evict using case 3 above). however, it was suggested that this traversal would only require reading of the dirty bit, which is a cheap read indicating that the caching policy is probably not unreasonably expensive (Jiwon).

3 Results

The proposed hybrid cache has a 15% energy reduction when compared with pure STT-RAM implementation and 31% higher throughput as compared to purely STT-RAM implementation.

3.1 Strengths

A noted strength of this paper was that it is very explicit about the experimental setup as well as the implementation of the proposed algorithms (Sam). This makes the results of the paper easy to replicate. Additionally, it's helpful if anyone decides to build off of their ideas in future work.

3.2 Weaknesses

Some weaknesses we identified in the paper were as follows:

1. They did not compare to full SRAM implementation, which would have been the upper bound on performance
2. They Do not note what kind of benchmarks they were using (write intensive versus read intensive) and how performance was affected by the benchmark type (Jiwon)
3. They did not mention their setup for simulating checkpointing. How long did they let the simulation warm up and run before doing the checkpointing procedure, and is that amount reasonable? (Jiwon)
4. They Do not evaluate the computational intensity of the checkpointing scheme. What kind of logic device or processor would be needed to implement this?

4 Alternate Approaches

There was a brief discussion about alternative approaches to implementing volatile to non-volatile caching and checkpointing algorithm. One proposed idea was to take inspiration from SSD life extending algorithms since these algorithms have similar characteristics to the pattern predictor (identifying MRU and LRU, several indicators, etc). However the SSD algorithm considers both pages and blocks when making decisions (pages are used for writing and blocks for erasing). Perhaps a similar consideration be beneficial here? (Quishen)

5 Suggested Reading

Citation 4 might be an interesting read as it pertains to how to optimize the back up process. It could be interesting to see how others propose this should be done. Can the data be compressed? How should the most important data to back up be determined if the time frame for backup is limited (Iris)