Toward Faster and More Efficient Training on CPUs Using STT-RAM-based Last Level Cache

Alexander Hankin, Maziar Amiraski, Karthik Sangaiah†, Mark Hempstead
Tufts University, Medford, MA, USA
†Drexel University, Philadelphia, PA, USA
hankin@ece.tufts.edu

I. Abstract

Artificial intelligence (AI), especially neural network-based AI, has become ubiquitous in modern day computing. However, the training phase required for these networks demands significant computational resources and is the primary bottleneck as the community scales its AI capabilities. While GPUs and AI accelerators have begun to be used to address this problem, many of the industry's AI models are still trained on CPUs and are limited in large part by the memory system. Breakthroughs in NVM research over the past couple of decades has unlocked the potential for replacing on-chip SRAM with an NVM-based alternative. Research into Spin-Torque Transfer RAM (STT-RAM) over the past decade has explored the impact of trading off volatility for improved write latency as part of the trend to bring STT-RAM on-chip. This is particularly STT-RAM is an especially attractive replacement for SRAM in the last-level cache due to its density, low leakage, and most notably, endurance.

II. Introduction and Motivation

Neural networks are often trained and deployed offline on a device with significant computational resources (or specialized hardware such as a GPU or TPU) and then inference decisions may be sent to a client device over a wireless network. The computational resources required for training is the major bottleneck in the way of ubiquitous, completely-distributed artificial intelligence (AI) on all kinds of devices. While most training is done using specialized hardware, some training is still done on the CPU, for example, in federated learning. For one to suggest that non-volatile memories (NVMs) may be helpful in training may seem counter-intuitive given the frequency of writes in training and the notorious write characteristics of NVMs; however, the training phase in particular actually has qualities that are well suited to NVMs, particularly Spin-Torque Transfer RAM (STT-RAM). That is, STT-RAM devices have a trade-off between retention time and write characteristics and AI training can be resilient to some errors in data retention.

It is well known that while STT-RAM (a prominent emerging NVM device for potential SRAM replacement in the last-level cache [1], [2]) has higher density and lower static power, the write energy and latency of STT-RAM is larger than SRAM. To reduce the undesirable effect, different solutions exist, e.g., early write termination [3], and hybrid caches [4], [5]. In addition to these, there is a different solution which approaches the problem from device level perspective: relaxing the non-volatility characteristic of STT-RAM in order to reduce STT-RAM write latency and energy consumption. The relationships that govern this relationship for the precessional switching regime of STT-RAM can be modeled by Equations 1–3 [6]. Recent work has leveraged the trade-off between volatility and write-latency to improve STTRAM-based memory performance [6]–[8].

$$\Delta \propto \frac{V \cdot H_K \cdot M_s}{T} \tag{1}$$

$$t_{retention} \propto Ce^{\Delta} \tag{2}$$

where:
Δ = thermal factor, V = volume, H_k = in-plane anisotropy field, M_s = saturation magnetization, and T = absolute temperature [K]

$$I_C(t_{write}) = A \cdot (J_{C0} + \frac{C}{t_{write}^{\gamma}}) \tag{3}$$

where:
I_C = current, A = cross sectional area of the free layer, J_{C0} = critical current density at zero temperature, C, γ = fitting constants

Training is a good candidate for exploiting the trade-off in latency and retention of STT-RAM because of its high reuse as well as its robustness to error–depending on where the errors occur–which can allow for more aggressive trade-offs in STT-RAM cell design (i.e., less volatility for faster speed). Additionally, training requires large working set sizes and is memory bound, so STT-RAM can enable significant increases in memory density at a low energy cost.

III. Exploration of a Low-Retention, STT-RAM-based Last Level Cache

To explore our hypothesis, we are developing a simulation test bed. We model a desktop CPU using an open source x86 simulator, Sniper [9], and the CPU model we use is based on an Intel Skylake processor running in turbo mode. We replace the SRAM-based LLC with an NVM-based model keeping the physical area the same. For our NVM LLC model, we select an STTRAM LLC model from [7], which includes models at different points on the trade-off between write latency and retention. In Sniper, we simulate regions of the training phase of different AI models, using C++ implementations, to measure performance and power, and to characterize LLC behavior. Figure 1 shows the distances between a cache line write and a subsequent read for a common fully-connected neural network. Most distances are between 1 and 100 ns. This is akin to the "natural" refresh rate of cache lines for the training phase.

To determine the optimal retention time for training a particular model, we measure the distribution of write-read distances in the LLC in the cache model in Sniper. We then form a statistical distribution for retention failure as a function of LLC block retention time. This allows us to estimate the number of retention errors that would occur for a particular STT-RAM model. The results from this process are shown in Table I for the fully-connected network used in Figure 1. Two different STT-RAM models were tested – one with a retention time of 1s and the other with 10ms. As can be seen from the table, the forward pass of the training phase is quite sensitive to STT-RAM retention time. Reducing retention from 1s to 1ms results in two orders of magnitude reduction in probability of a retention failure. On the other hand, backpropagation is not significantly sensitive to differences in retention time. This could be a consequence of the limited sample of retention times or just the nature of backpropagation. This should be explored further given that backpropagation is a larger percentage of training.

After characterizing the cache behavior in Sniper and estimating error probability, we look at how retention errors affect the accuracy of training. Based on the error distribution, we inject errors into the application code and re-train the model. Then we observe the accuracy that results from the injected errors and compare to an error-free training run. So far, we test injecting errors into the inputs of the model (and not the weights). For this, we turned to a well-known model: lenet. Here we used error probabilities of 10^{-7}, 10^{-6}, and 10^{-5}. The results of the study are shown in Figure 2. From the plot, we can observe that a 1e-05 error probability does not allow the model to learn if applied to layer 0 (conv, 1024), layer 1 (tanh, 4704), layer 2 (avg_pool, 4704), or layer 10 (FC, 120). To address this, we are exploring mitigation strategies which can be used to improve accuracy. One option is to use a hybrid LLC approach, which includes both SRAM and STT-RAM. When coupled with an effective mitigation, we believe that an intelligently designed STT-RAM-based LLC can improve performance and energy for the task of training with little to no impact on training accuracy compared to an SRAM-based LLC.

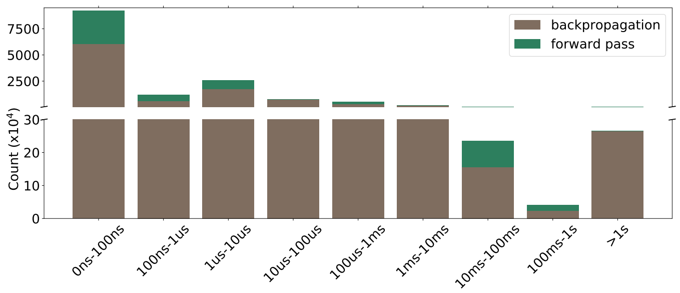

Fig. 1: Histogram showing distribution of distances between cache line insertion and subsequent read.

	Backpropagation	Forward Pass
P_{error} (STTRAM_1s)	0.003	0.00002
P_{error} (STTRAM_10ms)	0.002	0.002
P_{error} (SRAM)	0.000	0.000

TABLE I: Probability of retention failure for fully-connected network.

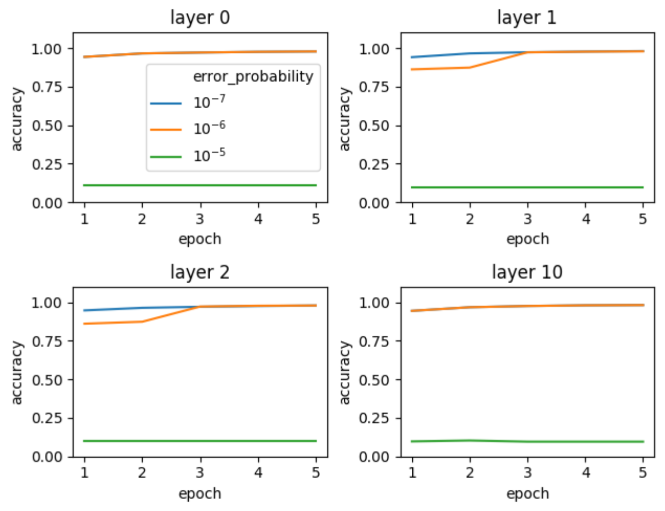

Fig. 2: Error tolerance of lenet to injected input errors. Error tolerance varies by layer.

References

[1] K. Korgaonkar *et al.*, "Density tradeoffs of non-volatile memory as a replacement for sram based last level cache," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 315–327.

[2] A. Hankin *et al.*, "Evaluation of non-volatile memory based last level cache given modern use case behavior," in *2019 IEEE International Symposium on Workload Characterization (IISWC)*, 2019, pp. 143–154.

[3] P. Zhou *et al.*, "Energy reduction for stt-ram using early write termination," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, Nov 2009, pp. 264–268.

[4] J. Ahn *et al.*, "Write intensity prediction for energy-efficient non-volatile caches," in *International Symposium on Low Power Electronics and Design (ISLPED)*, Sep. 2013, pp. 223–228.

[5] X. Wu *et al.*, "Power and performance of read-write aware hybrid caches with non-volatile memories," in *2009 Design, Automation Test in Europe Conference Exhibition*, April 2009, pp. 737–742.

[6] C. W. Smullen *et al.*, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, Feb 2011, pp. 50–61.

[7] A. Jog *et al.*, "Cache revive: Architecting volatile stt-ram caches for enhanced performance in cmps," in *DAC Design Automation Conference 2012*, June 2012, pp. 243–252.

[8] Z. Sun *et al.*, "Multi retention level stt-ram cache designs with a dynamic refresh scheme," in *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2011, pp. 329–338.

[9] T. E. Carlson *et al.*, “Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations,” in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov. 2011, pp. 52:1–52:12.