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

Alexander Hankin1, Maziar Amiriski1, Karthik Sangalia2, Mark Hemmstead1
1Tufts University, Medford, MA, USA
2Drexel University, Philadelphia, PA, USA
hankin@ece.tufts.edu, maziar@ece.tufts.edu, ks499@drexel.edu, mark@ece.tufts.edu

Abstract

Artificial intelligence (AI), especially neural network-based AI, has become ubiquitous in modern 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. Breakthroughs in NVM research over the past couple of decades have unlocked the potential for replacing on-chip SRAM with an NVM-based alternative. STT-RAM is an especially attractive replacement for SRAM in the last-level cache due to its density, low leakage, and most notably, endurance. Research into Spin-Torque Transfer RAM (STT-RAM) has explored the impact of trading off volatility for improved write latency.

1. Introduction

- Neural networks are often trained and deployed offline and inference decisions sent to a client device over a wireless network.
- The training phase is the major bottleneck in the way of ubiquitous, completely-distributed artificial intelligence (AI) on all kinds of devices.
- Some training is still done on the CPU, for example, in federated learning.
- Non-volatile memories (NVMs) may seem counterintuitive given the frequency of writes in training and the notorious volatility characteristic of STT-RAM.

2. 2. Motivation

- STT-RAM-based cache [1], [2] has higher density and lower static power than SRAM, but write energy and latency is larger.
- Relax the non-volatility characteristic of STT-RAM in order to reduce STT-RAM write latency and energy consumption.
- The relationships that govern this trade-off can be modeled by Equations 1-3 [6]:

\[ \Delta = k \cdot V \cdot \text{mc} \]

\[ t_{\text{retention}} = \frac{1}{C} \cdot e \]

\[ I_2(\text{area}) = A \cdot (J \cdot C + \frac{1}{C}) \]

where:
- \( \Delta \) = thermal factor
- \( V \) = voltage
- \( \text{mc} \) = in-plane anisotropy field
- \( M_s \) = saturation magnetization
- \( T \) = absolute temperature [K]
- \( I_2 \) = current density at zero temperature, and
- \( C \) = fitting constants
- Training is a good candidate for exploiting this tradeoff because of the high reuse as well as its robustness to error.
- Also, 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.

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

- Using an open source x86 simulator, Sniper [9], we created a simulation testbed of a CPU model based on a 14nm Intel Skylake processor running in turbo mode. We replace the SRAM-based LLC with an NVM-based LLC [7] keeping the physical area the same.
- Figure 2 shows the distances between a cache line write and a subsequent read for a recent fully-connected neural network. Most distances are between 1 and 100 ns.
- To determine optimal retention time, we form a statistical distribution for retention failure as a function of LLC retention time (Table I).
- One STT-RAM model with a retention time of 1s and the other 10ms. With the farthest 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.
- To see how retention errors affect accuracy of training, we inject errors into the model

We inject errors into the activations of lenet with error probabilities of 10\(^{-2}\), 10\(^{-4}\), and 10\(^{-6}\).

5. References