MLWeaving: A One-Size-Fits-All System for Any-precision Learning

Zeke Wang, Kaan Kara, Hantian Zhang, Gustavo Alonso, Onur Mutlu, Ce Zhang

Problem: Linear model training using low-precision SGD.

Existing Approach:
1. One hardware design for each precision.
2. One quantized dataset for each precision.

Our Approach (MLWeaving):
One hardware design and one memory layout for any precision.

MLWeaving: Software/Hardware Co-design

MLWeaving memory layout (software)
1. Original full-precision fixed-point table for training dataset a
2. MLWeaving memory layout for training dataset a

MLWeaving arithmetic (hardware)

Hardware: an Intel Broadwell CPU (14 cores, 35MB LLC, 60 GB/s memory bandwidth) and an Intel Arria 10 FPGA (directly access the CPU memory via one QPI and two PCie, with memory bandwidth: 15GB/s).

Dataset: Epsilon (40,000 samples, 2000 features).

Hogwild (ModelAverage): state-of-the-art parallel implementations of SGD on CPUs, using 14 cores, AVX2 and 8-bit dataset.

Experiment

Findings:
1. MLWeaving can roughly achieve linear speedup (time or memory traffic), when a lower number of bits is used, as shown in Figures a, b.
2. MLWeaving on an FPGA can achieve 11X speedup over its CPU rivals in Figure c.

Stochastic Gradient Descent (SGD):

\[
\text{For } e = 1 \text{ to } E \text{ do} \\
\text{For } i = 1 \text{ to } N \text{ do} \\
\quad x = Q_s(\alpha_i) \cdot \hat{x}; \\
\quad g = \gamma \cdot df(ax, b); \\
\quad x = x - g; \\
\]

where:
- \(\hat{x}\): model
- \(g\): gradient
- \(\Gamma\): learning rate
- \(df\): derivative of loss function

(a) Time vs. Precision
(b) Memory traffic vs. Precision
(c) MLWeaving vs. CPU rivals