CS 5220: Final Project

Hogwild! and Buckwild!

Alex Renda (adr74), Matthew Li (ml927), Matt Gharrity (mjg355)

November 28, 2017

CS 5220: Applications of Parallel Computers

Stochastic Gradient Descent (SGD)

Looking for a vector (ML model) that minimizes a loss function.

- **Gradient descent**: minimize the loss function by taking small steps in the direction of its gradient.
- **Stochastic gradient descent**: gradient descent, but randomly sample a small subset of training data for computing the loss. Makes things faster and avoids local minima.

Problem: hard to parallelize. SGD is inherently sequential because each update depends on the last.

Hogwild!

“A simple strategy for eliminating the overhead associated with locking: run SGD in parallel without locks”

Why it works: updates to the vector are *sparse*: only a few entries are updated at a time. Thus threads will rarely overwrite each other’s work. Besides, machine learning algorithms already approximate solutions.

Memory model

Still need *some* form of synchronization—otherwise threads might cache the vector and never see each other's updates!

The original HOGWILD! paper assumes memory consistency that can be implemented with OMP. Updates to weight vector must be done in a critical section, otherwise memory is considered local. This is weaker than sequential consistency.

We will experiment with weakening the critical section requirement is possible, by flushing after updating to the weight vector instead of placing it in a critical section.

Buckwild!

Using 32-bit floating point numbers is slow.

We can instead use 8-bit integer numbers with rounding. Using 8-bit integers allows Buckwild! to take advantage of SIMD instructions for integers on CPUs, further improving performance.

This introduces additional noise into the system, but does not affect convergence rate.

We use `https://github.com/google/gemmlowp`.

Experiment

- Run logistic regression on MNIST Image Dataset using sequential SGD, HOGWILD!, and BUCKWILD!.
- Report training loss and testing error with respect to iterations and wall clock time; try to verify the claims made in the original papers.
- Testing done on the totient cluster, compiled with `icc`.