EventGraD: Event-Triggered Communication in Parallel Stochastic Gradient Descent

Soumyadip Ghosh and Vijay Gupta

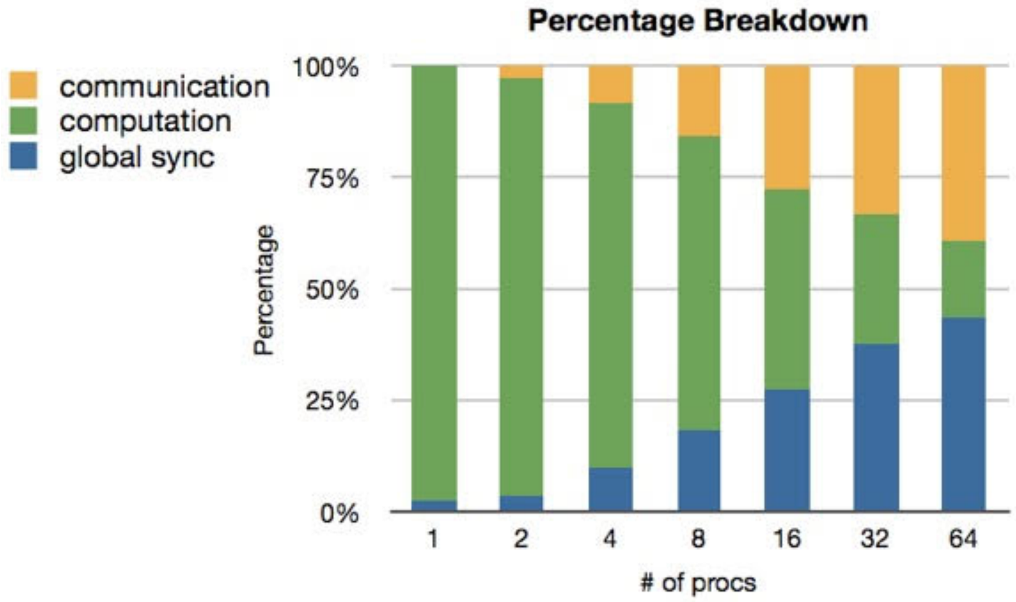
University of Notre Dame

Machine Learning in HPC Environments (MLHPC) workshop

November 2020

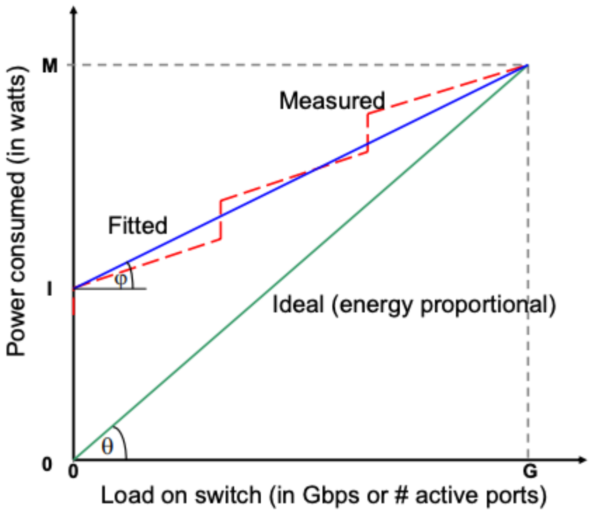
Communication in Parallel Machine Learning slow!

- Communication between processing elements takes more time!

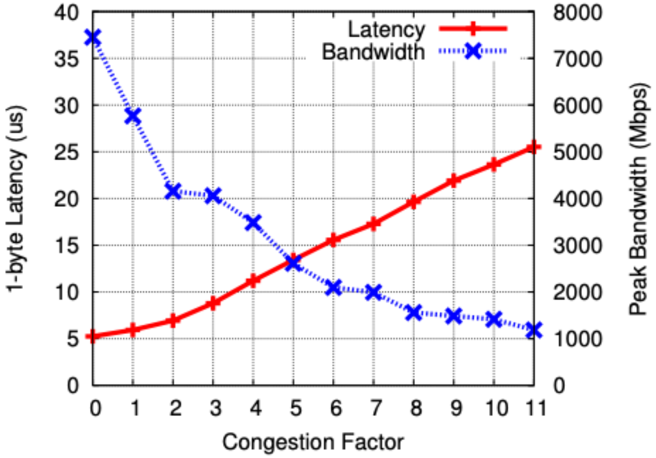


Other Communication Overhead

- Communication of messages consumes energy*



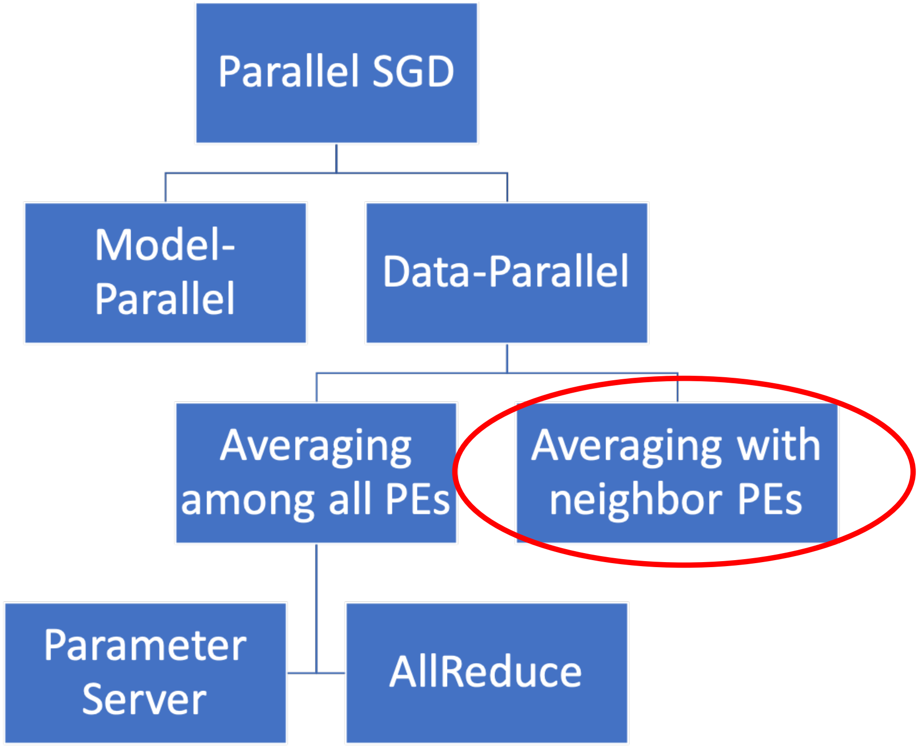
- Communication may lead to congestion in HPC interconnects#

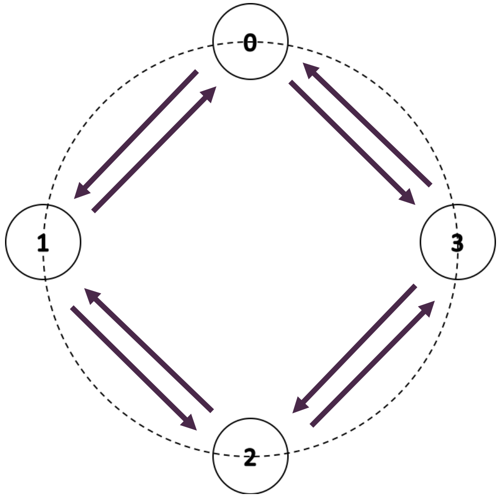


*P. Mahadevan et al, "A Power Benchmarking Framework for Network Devices", Springer, 2009.
#T. Hoefler et al, "Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks, IEEE, 2008.

Communication in Parallel Neural Network Training

- Strategies of parallelizing neural networks





X. Lian et al. "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent." *Advances in Neural Information Processing Systems*. 2017.

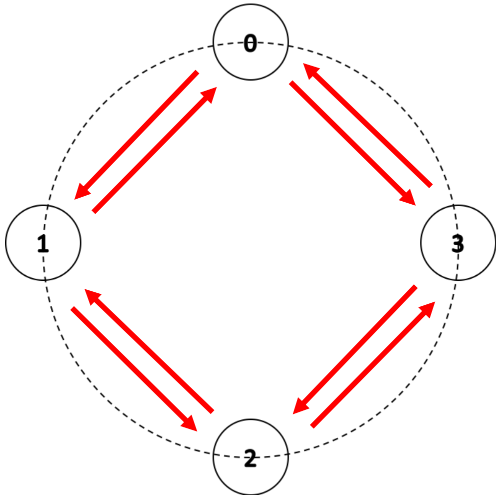
Existing Approaches vs Our Approach to Reduce Communication

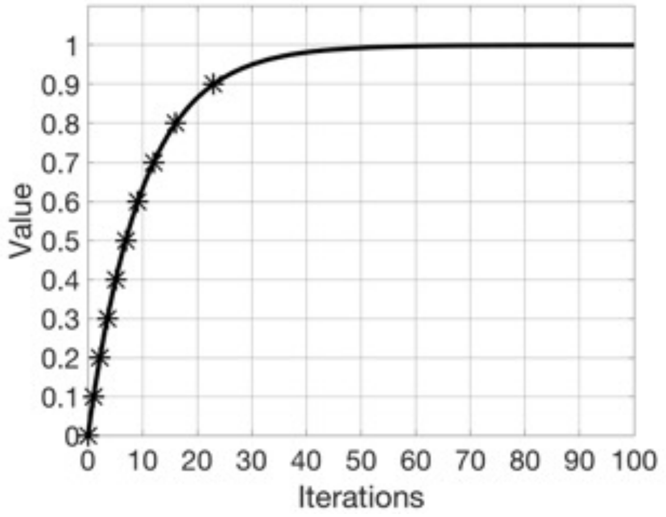
- Concentrated on averaging among all processors
- Parameter Server approach
  - Hogwild! (Recht et al. 2011)
  - Elastic Averaging SGD (Zhang et al. 2015)
- AllReduce approach
  - Quantization in gradients – One bit (Seide et al. 2014), Threshold (Strom et al. 2015), Mixed (Dryden et al. 2016)
  - Sparsification in gradients – Top-K significant (Alistarh et al. 2018), Deep Gradient Compression (Lin et al. 2018)
  - Changed precision – Low (Gupta et al. 2015), Mixed (Micikevicius et al. 2018)

We focus on averaging with neighbors and communicate in an event-triggered fashion

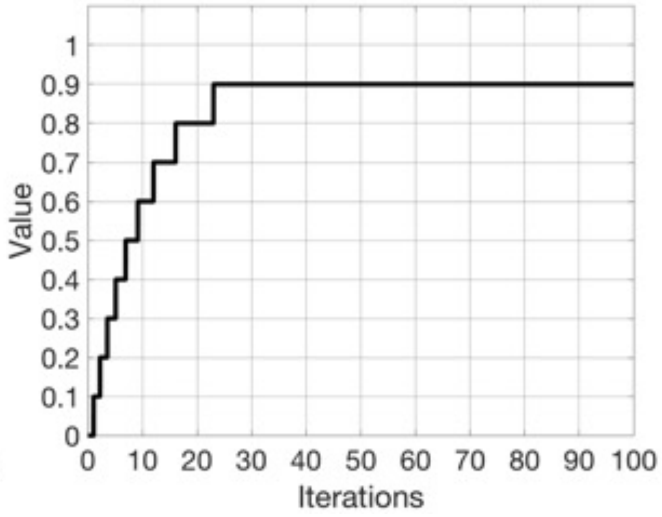
Our Proposed Algorithm

- Communicate model parameters (weights and biases) in events only when necessary





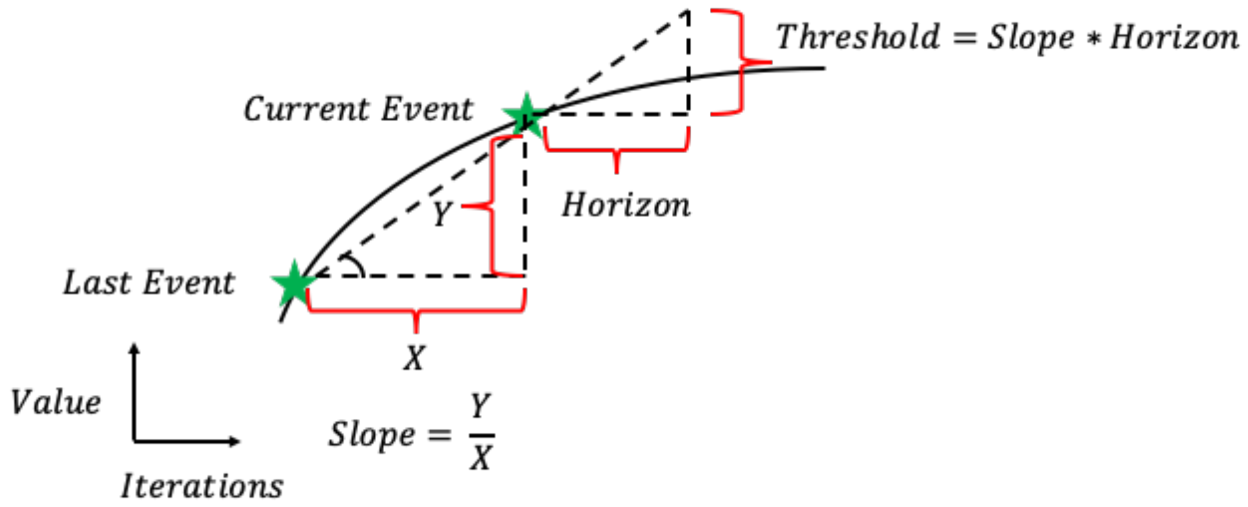
Sender sends a parameter only when the change in its norm exceeds a threshold



Receiver continues computation using the last received values

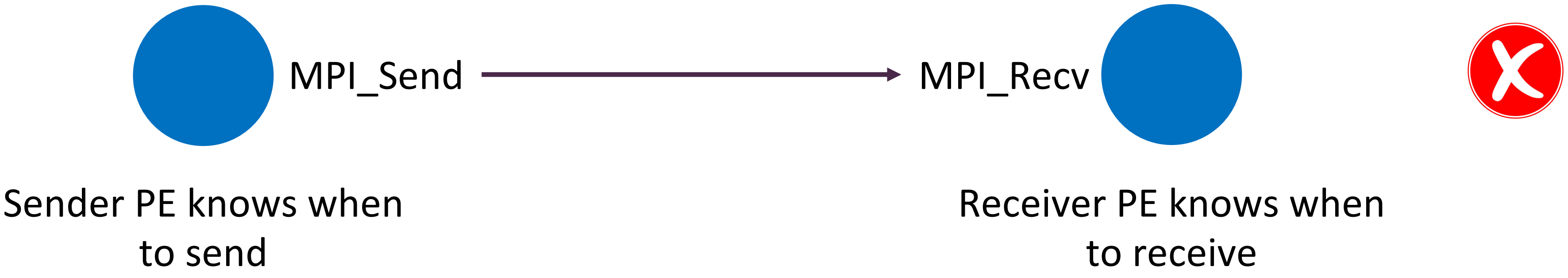
How to choose the threshold?

- Choosing the threshold for communication is a design problem
- Simplest option is to choose a constant threshold
- Better option is an adaptive threshold based on the slope of the parameters



Implementation : Need for One-Sided Communication

- Event of communication is triggered based on parameters at the sender
- Receiver is not aware when event is triggered at sender



MPI Two-sided Communication

MPI One-sided Communication

Implementation : Framework

- Distributed Modules in PyTorch, TensorFlow, CNTK, Horovod provide either the Parameter Server or AllReduce approach

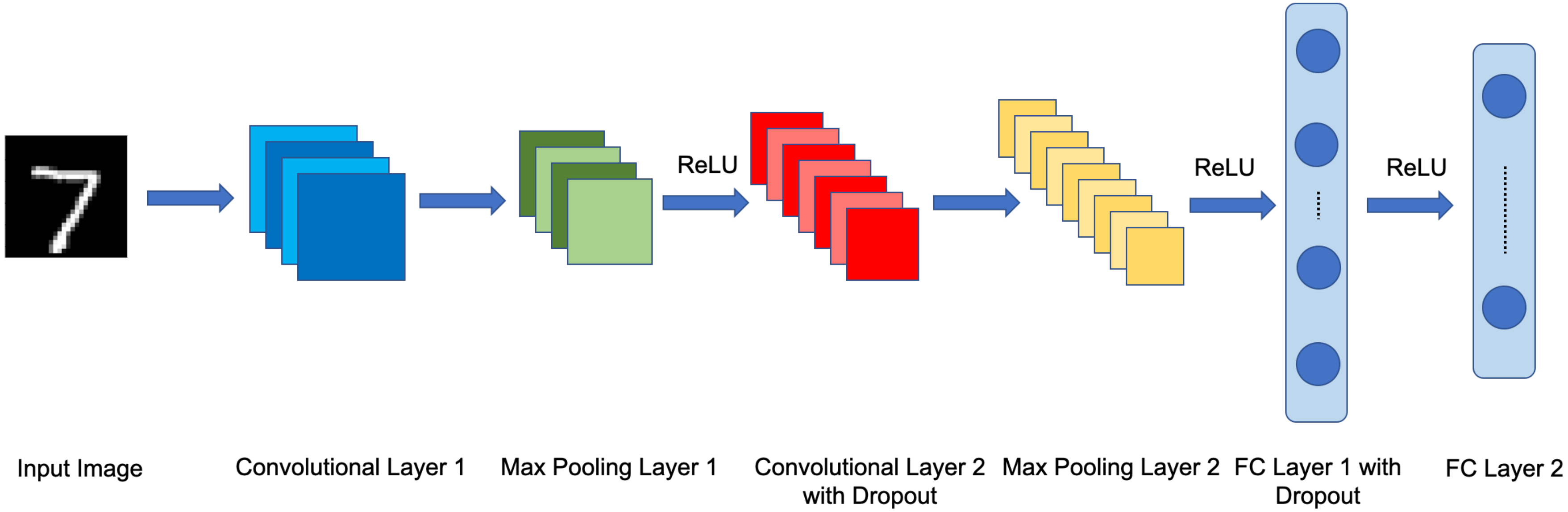
- No module supports training involving averaging with just neighbors

- We implement using primitives from PyTorch and MPI

- PyTorch has a C++ frontend (Libtorch) which we combine with MPI one-sided functions

Experiments

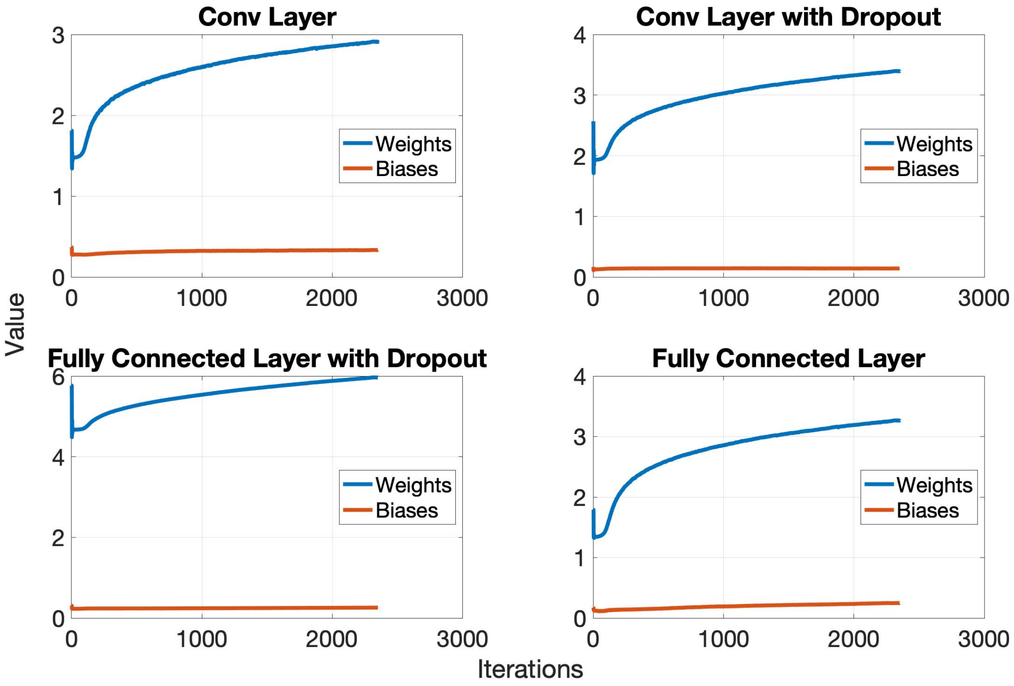
- We train a popular convolutional network on the MNIST dataset



Libtorch MNIST example. https://github.com/pytorch/examples/blob/master/cpp/mnist/mnist.cpp. Accessed: 07-11-2020.

Evolution of Norm of Parameters

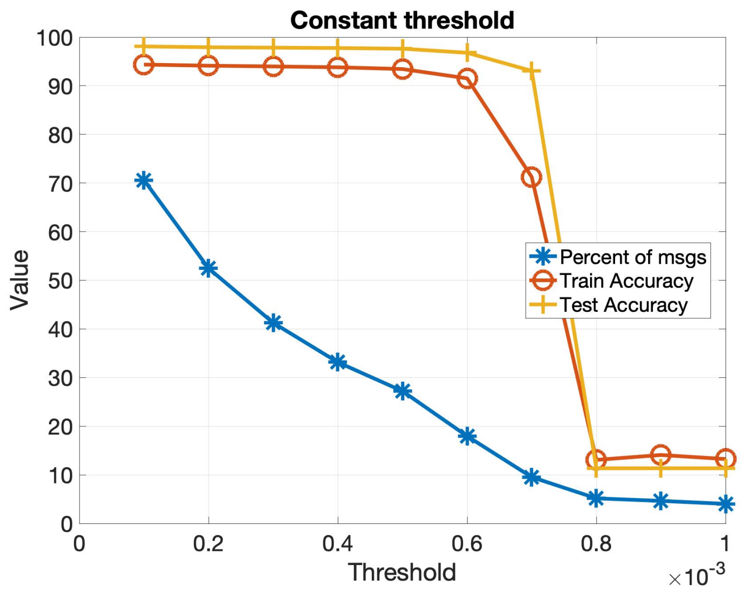
- Euclidean norm of parameters in our model over iterations in one processor



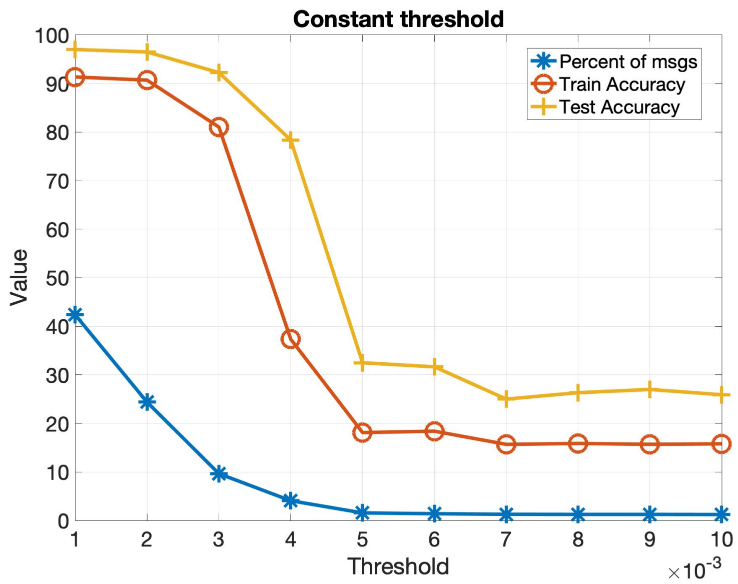
- After initial oscillations, weights have a gradual change while bias stays almost flat

Evaluation using a constant threshold

- Choosing a constant threshold for triggering events



Kernel size of convolutional layers = 5

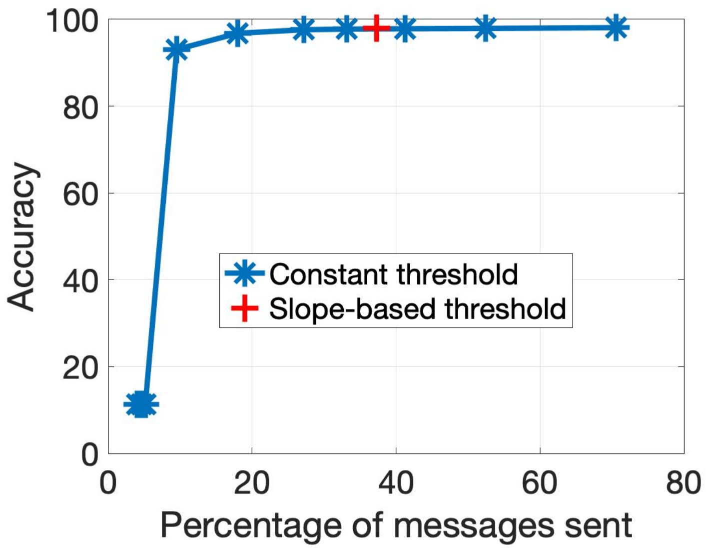


Kernel size of convolutional layers = 3

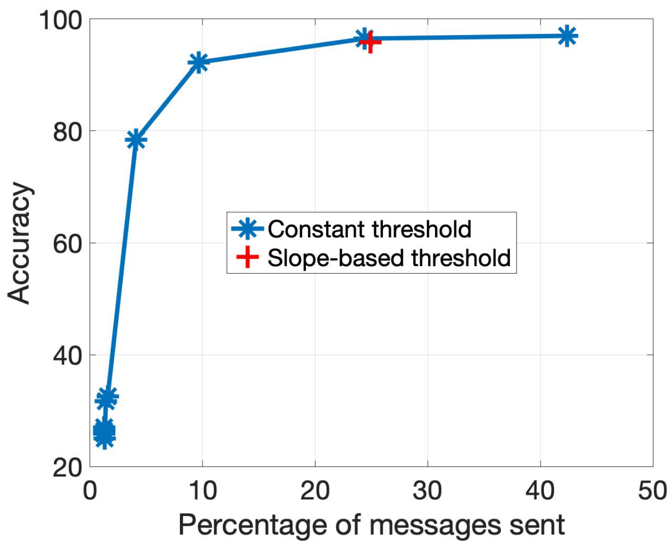
- Changing the model changes the range of acceptable thresholds which has to be found before training

Comparison between constant and adaptive threshold

- Slope-based adaptive threshold does not require extensive tuning



Kernel size of convolutional layers = 5



Kernel size of convolutional layers = 3

Using horizon = 1 for the adaptive threshold for both

- However adaptive threshold might be sub-optimal

Future Work

- We need to do more experiments of larger models on larger datasets
  - Resnet-50 on ImageNet
- Extend code to GPUs
- Investigate theoretical properties such as rate of convergence and optimal way to choose the adaptive threshold

Source Code and Secondary Objectives

- Check https://github.com/soumyadipghosh/eventgrad
- Combines PyTorch C++ API and MPI to implement parallel machine learning in general
- Also implemented AllReduce based training and decentralized training with neighbors
- Add to PyTorch examples repository - https://github.com/pytorch/examples/pull/809
- PyTorch C++ API is still in beta – coverage of models and datasets is quite limited!

SC20

Summary

- Propose a novel communication algorithm for decentralized machine learning based on events
- Event-Triggered Communication reduces the number of messages, thereby saving on communication time and energy
- Emphasize on an adaptive threshold dependent on the slope of parameters
- Can be extended to training algorithms other than SGD and model-parallel configurations

http://sites.nd.edu/sghosh/ sghosh2@nd.edu