Communication Quantization for Data-parallel Training of Deep Neural Networks

MLHPC 2016

November 14, 2016

Nikoli Dryden[1,3], Tim Moon[2,3], Sam Ade Jacobs[3], Brian Van Essen[3]

[1] University of Illinois at Urbana-Champaign
[2] Stanford University
[3] Lawrence Livermore National Laboratory

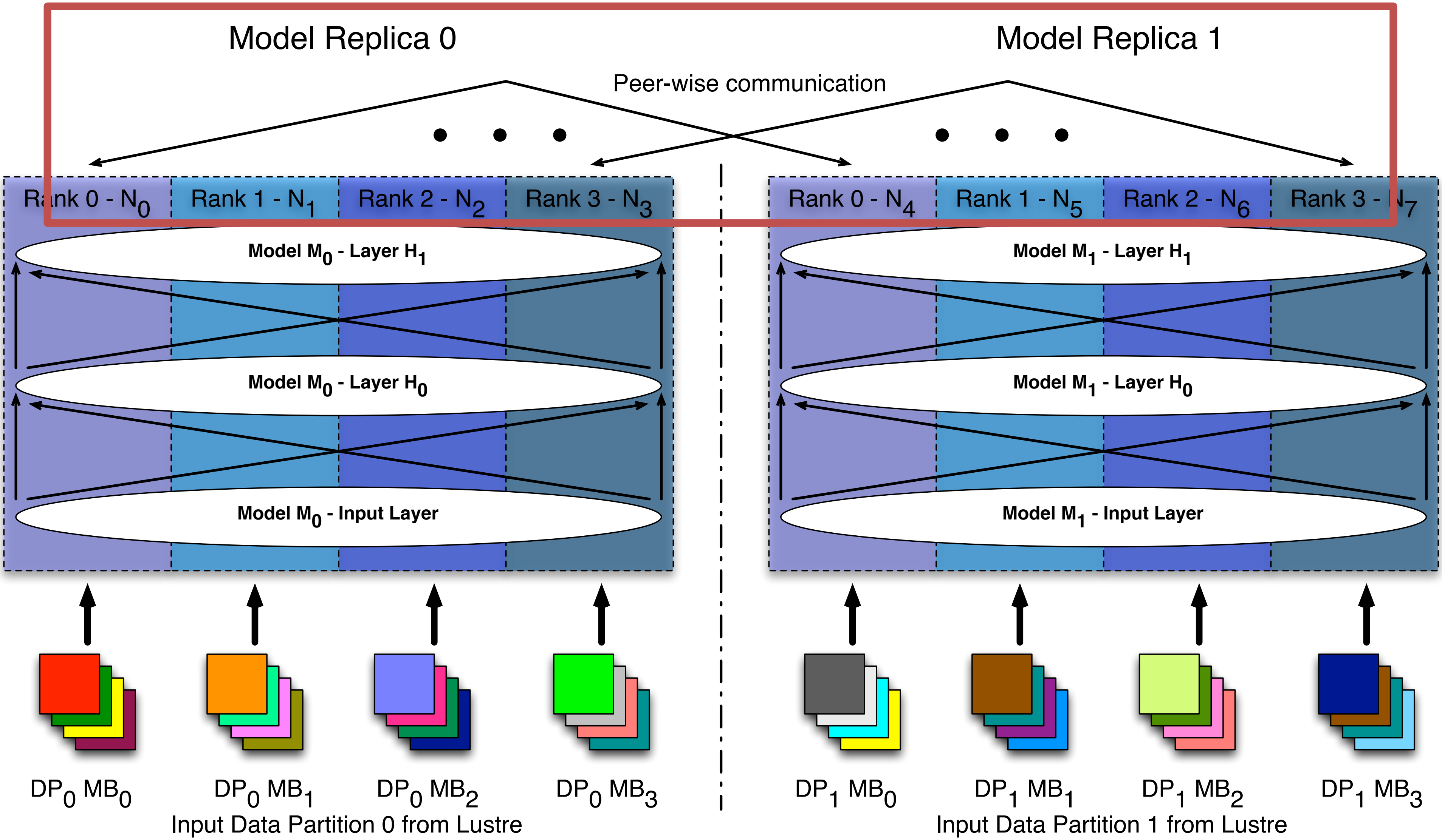
LLNL-PRES-708391
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Motivation

- Training DNNs is very intensive
- Datasets continue to become larger and larger
- Let's try to take advantage of HPC resources

Summary

- Quantize gradient updates and use a custom communication algorithm
  - Reduces bandwidth during data-parallel training
- Outperform baseline for large layers (1.76x)
- Code available: https://github.com/LLNL/lbann

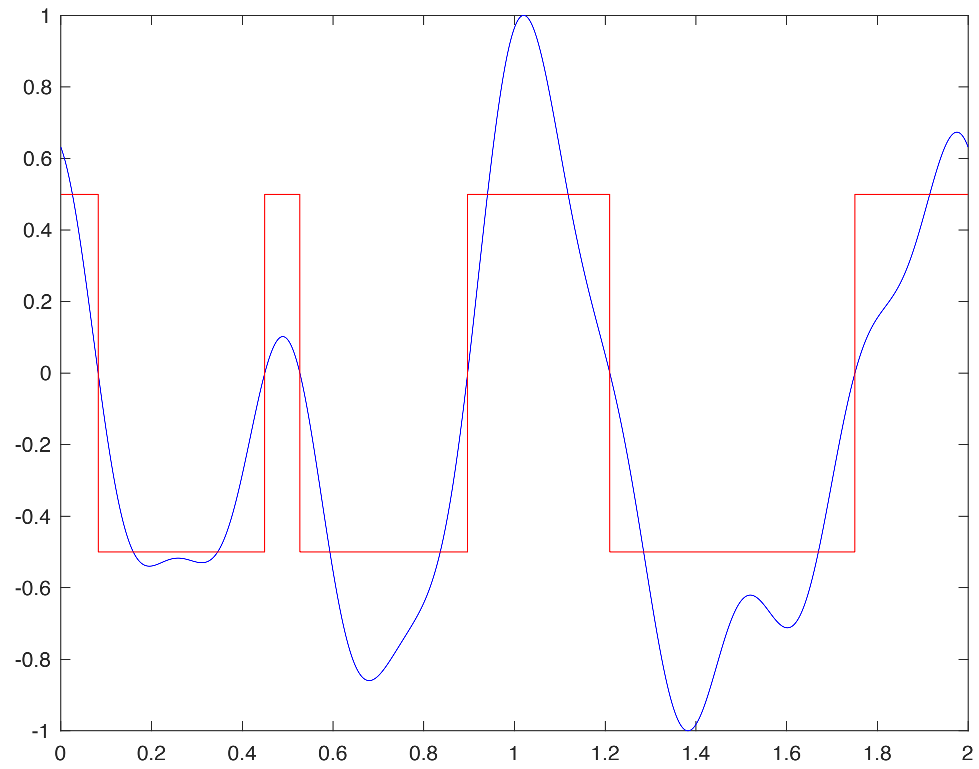

Model Replica 0
Model Replica 1
Peer-wise communication
Rank 0 - N0
Rank 1 - N1
Rank 2 - N2
Rank 3 - N3
Rank 0 - N4
Rank 1 - N5
Rank 2 - N6
Rank 3 - N7
Model M0 - Layer H1
Model M0 - Layer H0
Model M0 - Input Layer
Model M1 - Layer H1
Model M1 - Layer H0
Model M1 - Input Layer
DP0 MB0
DP0 MB1
DP0 MB2
DP0 MB3
Input Data Partition 0 from Lustre
DP1 MB0
DP1 MB1
DP1 MB2
DP1 MB3
Input Data Partition 1 from Lustre

Why is this hard?

- Communication-computation imbalance
  - You spend more time communicating than doing useful work!
  - Bandwidth-dominated regime
- Existing work more focused on heterogeneous cloud infrastructure, not HPC

Quantization

- Map a large set of values to a smaller set
- Quantized data is reconstructed using a pre-computed dictionary
- Introduces some amount of quantization error
- In our case: map 32-bit floats (gradient updates) to 1 bit



Quantization algorithms

- Trade increased (local) computation for reduced data movement
- Existing approaches:
  - One-bit quantization [F. Seide et al. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. INTERSPEECH 2014]
  - Threshold quantization [N. Strom. Scalable distributed DNN training using commodity GPU cloud computing. INTERSPEECH 2015]
- New: Adaptive quantization

One-bit quantization

- Aggressively quantize every update to 1 bit
- Compute column-wise means of non-negative/negative gradient updates
- Gradient updates $\geq 0 \rightarrow \mu^{+}$
  Gradient updates $< 0 \rightarrow \mu^{-}$
- Encoded as a 0 or 1 bit, data volume reduced 32x with packing
- Introduces *error feedback* to correct quantization error

One-bit quantization: visual

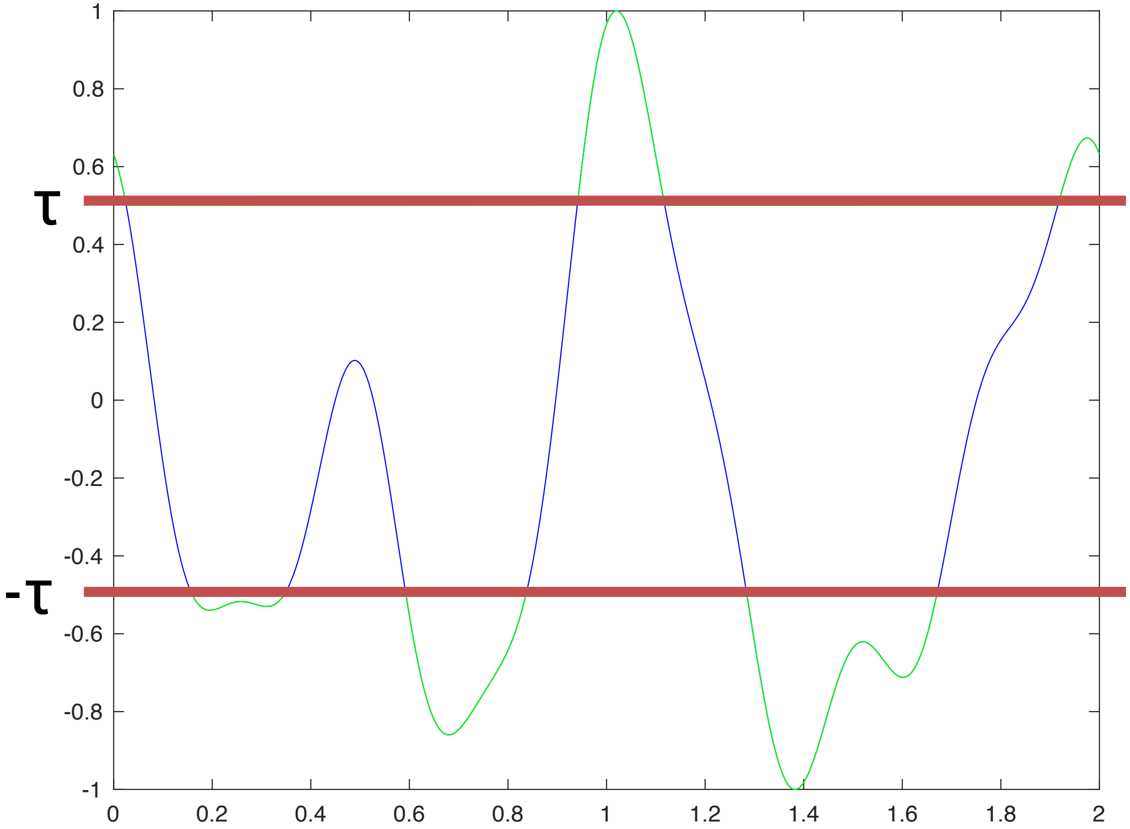
Error feedback

- Aggressive quantization introduces error
  - Ignoring it leads to poor models or divergence
- Instead retain the quantization error locally and add it to the gradient updates in the next mini-batch before quantization
- Ensures the full gradient signal is—eventually—used, just over multiple updates

Threshold quantization

- Instead of sending every update, send only the largest
- User chooses a fixed threshold τ in advance
- Gradient updates ≥ τ → τ
  Gradient updates ≤ -τ → -τ
- Encoded as 0 or 1 bit
- Error feedback used to reduce error
- Other updates are fed into error feedback but not used
- Updates are now sparse: each quantized gradient is sent as a 31-bit index and a 1-bit quantized value

Threshold quantization: visual



Adaptive quantization

- Motivation
  1. Threshold quantization can be fast with a good τ
  2. … But τ is hard to choose in practice
  3. … And τ/-τ are not great reconstruction values
  4. One-bit quantization seems to be more consistent
  5. And has no parameters to choose
- Adaptive quantization tries to get the best of both worlds

Adaptive quantization

- User chooses a fixed proportion of updates to send
- Algorithm determines the appropriate thresholds τ^+, τ^- to achieve this
  - Then determines the mean μ^+ of the updates greater than τ^+ and the mean μ^- of the updates less than τ^-
- Gradient updates $\geq \tau^+ \rightarrow \mu^+$
  Gradient updates $< \tau^- \rightarrow \mu^-$
- Error feedback used to reduce error
- Updates are sparse and use the same format as threshold quantization

Additional optimizations

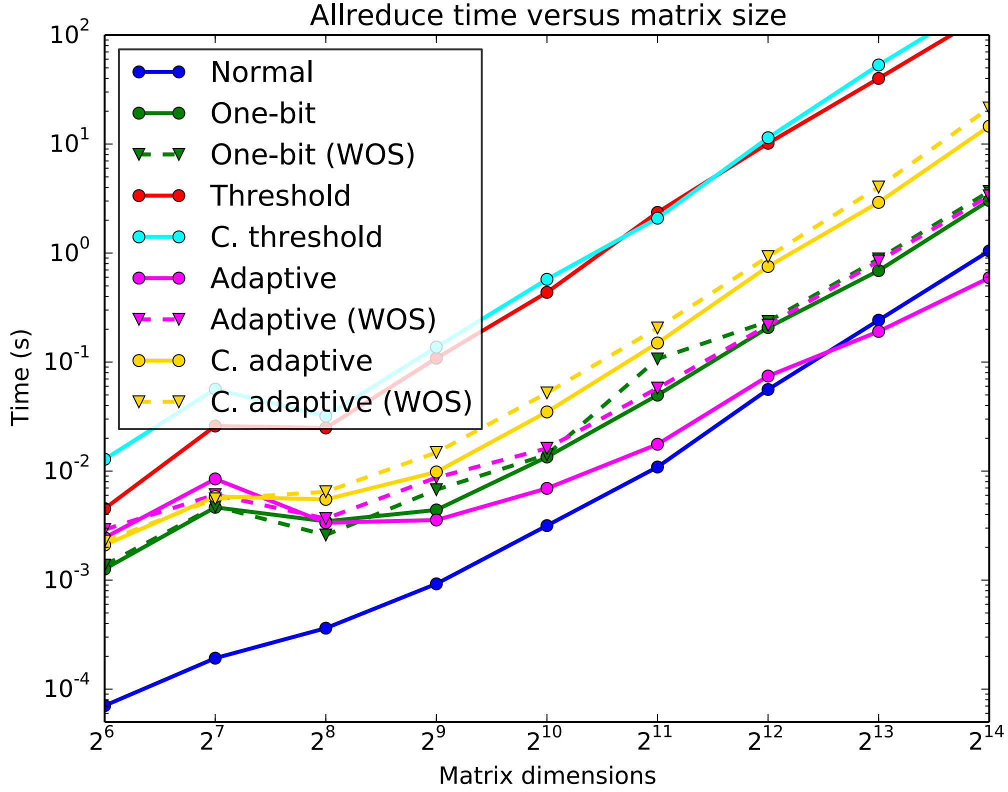
- One-bit and adaptive:
  - Approximate some computations using random sampling
- Threshold and adaptive:
  - Delta and Golomb-Rice coding for additional compression to reduce data volume

Allreduce

- Key communication operation for updates
- MPI_Allreduce is good in theory, but not in practice
  - Uses default algorithm with custom datatypes
  - Troublesome to associate reconstruction dictionaries
  - Does not handle changing data sizes well
- Implement using pairwise exchange reduce-scatter then ring-based allgather
  - $O((^{p-1}/_{p})n\beta)$ versus $O(n\log(p)\beta)$ (default) bandwidth
  - [R. Thakur et al. "Optimization of collective communication operations in MPICH." IJHPC, 2005]

Quantization benchmark

- Uniformly random square matrices
- 128 nodes, 2 processes/ node
- Simulates gradient updates with 128-way data parallelism
- Adaptive quantization superior for large matrices: 1.76x faster for largest matrix

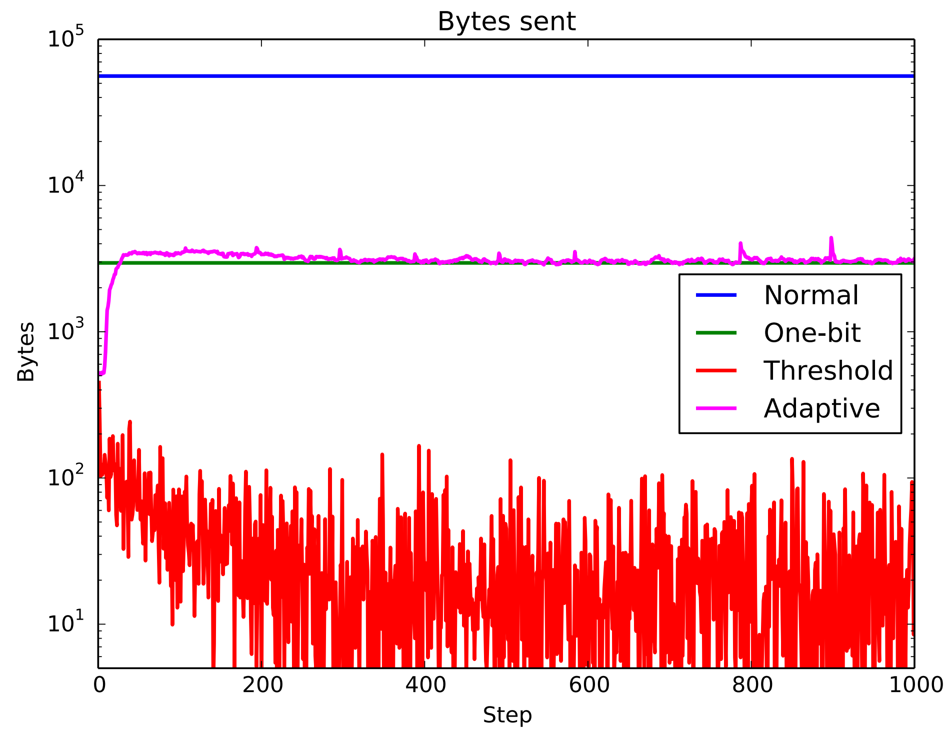


Test setup

- MNIST handwritten digit dataset
- 3 4096-neuron fully-connected hidden layers
  - ReLU activations
  - Adagrad optimizer
- 16 nodes, 192 ranks
  - 4-way data parallelism
  - 48-way model parallelism

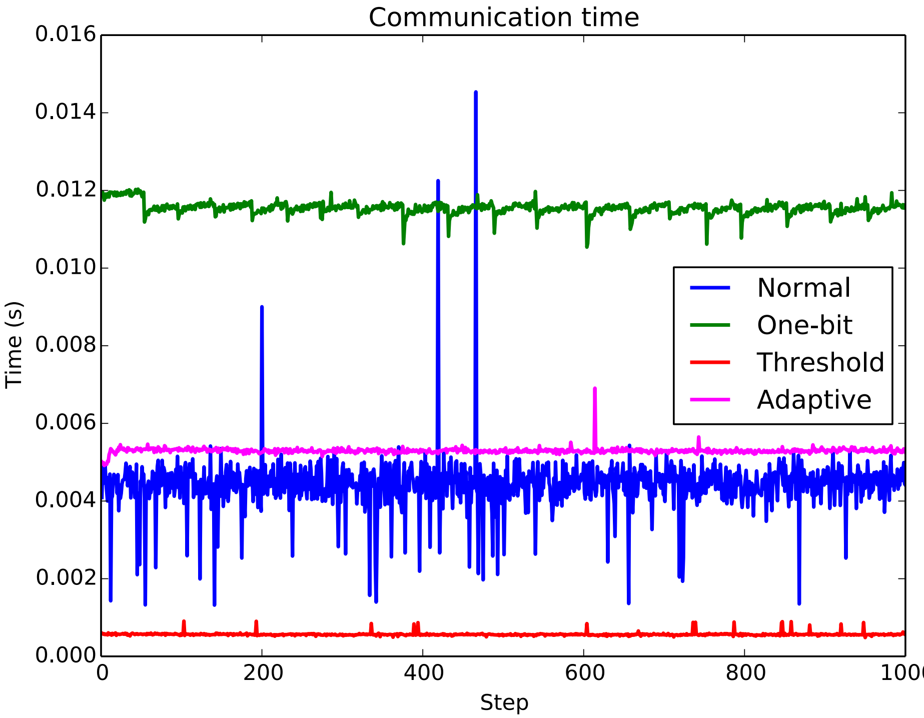
Data volume reduction

- Bytes sent in each mini-batch during training
- Adaptive quantization closely follows one-bit quantization (expected)
- Threshold quantization is degenerate and sends very little data (using best τ we found)



Communication time

- Total time spent in the allreduce in each mini-batch
- Times in line with the quantization benchmark
- Threshold quantization sends very little data, so is much faster



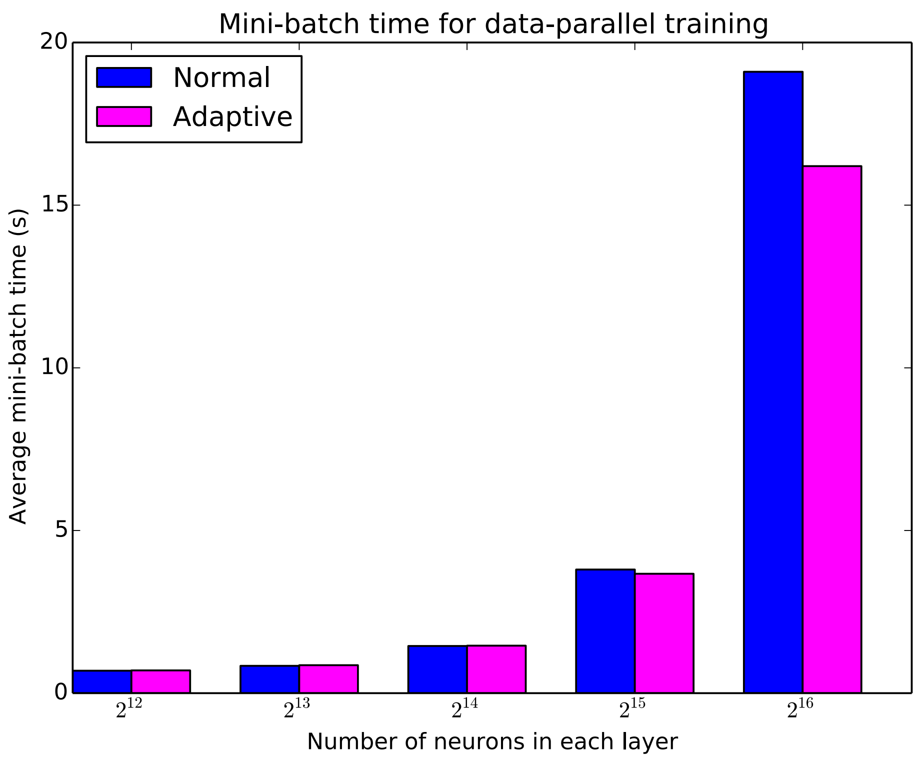
Accuracy

- Important that quantization does not degrade accuracy
- Normal, one-bit, and adaptive quantization lead to comparable accuracies
- Threshold accuracy is comparable to that of a single model replica

	Test accuracy (%) after 20 epochs
Normal	98.51
One-bit	98.49
Threshold	98.12
Adaptive	**98.53**

Layer scaling

- Increase neurons in each layer: 1.18x faster for largest layer
  - Validates the quantization benchmark in a more realistic training situation
  - Adaptive quantization has the advantage for larger problems



Discussion

- Bandwidth reduction through quantization and custom communication routines help scale data parallel training
- Adaptive quantization is fast and easy to tune
- Next steps:
  - Further optimization
  - Convolutional layers and GPUs
  - Larger datasets (ImageNet)

Lawrence Livermore
National Laboratory