High Performance Computing 2015

JIAN GUO

TOKYO INSTITUTE OF TECHNOLOGY

DEPT. OF MATHEMATICAL AND COMPUTING SCIENCES

MATSUOKA LAB.

Reviewed Paper

Title: FireCaffe: near-linear acceleration of deep neural network training on compute clusters

Authors: Forrest N. Iandola, Khalid Ashraf, Mattthew W. Moskewicz, Kurt Keutzer

DeepScale and UC Berkeley

[http://arxiv.org/abs/1511.00175]

Abstract

- FireCaffe, which successfully scales deep neural network training across a cluster of GPUs.
  - Reduce communication overhead
  - while not degrading the accuracy of the DNN models

- Three key pillars
  - Network hardware that achieves high bandwidth between GPU servers – Infiniband or Cray interconnects
  - Present a communication algorithm named reduction trees, which is more efficient and scalable than the traditional way
  - Optionally increase the batch size to reduce the total quantity of communication during DNN training, and we identify hyperparameters that allow us to reproduce the small-batch accuracy while training with large batch sizes.

Outline

1. Introduction and Motivation
2. Accelerating DNN Research and Development
3. Preliminaries, terminology and parallelism strategies
4. Choosing DNN architectures to accelerate
5. Methodology
6. Evaluation and Results
7. Conclusions

My Impression

Introduction and Motivation

- A variety of new deep neural network (DNN) architectures such as GoogleNet [1], Network-in-Network [2], VGG [3] and AlexNet [4] have been developed at a rapid pace for improvements in **accuracy.**
- What is the bottleneck in developing of new architectures?
- All of which operate near the theoretical peak computation per second achievable GPUs in single node.
- "Training time is a key challenge at the root of the development of new DNN architectures." – by Jeffrey Dean of Google in his keynote address at the 23rd International Conference on Information and Knowledge Management. [5]

[1]C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. arXiv:1409.4842, 2014.
[2] M. Lin, Q. Chen, and S. Yan. Network in network. arXiv:1312.4400, 2013.
[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014.
[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In NIPS, 2012.
[5] J. Dean. Keynote: Large scale deep learning. In ACM Conference on Information and Knowledge Management (CIKM), 2014.

Why we need to accelerate DNN Research and Development

Due to long training times, these companies are facing serious delays in bringing DNN-based solutions to market.

Faster DNN training would enable this and other reinforcement learning applications to move toward real-time.

Contributions of this Paper

- Presented FireCaffe--Scaling deep neural network training across a cluster of GPUs
- DNN training across a cluster of 32 GPUs with speedups of more than 20x compared to a single GPU.
- Scaling up DNN training by the philosophy to balance computation and communication.
- A number of hardware and software design choices to lower communication time.

Preliminaries and terminology

- Deep neural network is comprised of iterating between two phases:
  - Forward phase: a batch of data items (e.g. images) is taken from the training set, and the DNN attempts to classify them.
  - backward phase: which consists of computing gradients with respect to the weights (∇W) and gradients with respect to the data (∇D)

Preliminaries and terminology

- Equation1 shows total size (in bytes) of the weights in all convolutional and fully-connected layers

ch is the number of channels, numFilt is the number of filters, filterH is the filter height, and filterW is the filter width.

$$|W| = \sum_{L=1}^{\#layers} ch_L * numFilt_L * filterW_L * filterH_L * 4 \quad (1)$$

floating-point number is 4 bytes

$$|D| = \sum_{L=1}^{\#layers} ch_L * numFilt_L * activationW_L * activationH_L * batch * 4 \quad (2)$$

- Equation2 presents the size of activations produced by all layers, activationH is the activation map height, activationW is the activation width, and batch is the batch size

Parallelism strategies

- Data parallelism: worker (e.g. GPU) gets *a subset of the batch*, workers communicate by exchanging weight gradient updates ∇W
- Model parallelism: worker gets *a subset of the model parameters,* workers communicate by exchanging data gradients ∇D and exchanging activations D.

Q1. Why we may get incorrect results after parallelization?

- In shared-memory systems, processors execute asynchronously may lead to “nondeterminism” which will cause different outputs from a given input. Especially when two threads attempt to access the same resource, and it can result in an error called "a race condition".

  The second potential problem is "thread safety", Functions that were written for use in serial programs can access static variables, and this use in a multithreaded program can cause errors. Such functions are not thread safe.

- In distributed-memory systems, we use Message-Passing Interface(MPI) to program. When using MPI_Recv, we may "hang" the process. If a call to MPI_Send blocks and there is no matching receive, then the sending process will "hang". On the other hand, a call to MPI_Send is buffered and there is no matching receive, then the message will be lost.

Q2. The batch size

1000000 images, every image has 2000x2000 pixels and batch size is 1000.

In this case, 1000000 images will be divided by 1000 (batch size). The whole work is divided into 1000 times to complete, and the program will process 1000 images in one time. There is no relationship between the pixels and the batch size.

Data parallelism at typical batch sizes (1024)

- Most DNN models are consist primarily of convolution layers.
- For convolutional models, data parallelism is typically preferable because it requires less communication ,$|\nabla W|$ is much smaller than $|\nabla D|$ at typical batch sizes (1024).

Table 1. Volumes of data and computation for four widely-used DNN architectures. The batch size impacts all numbers in this table except for $|W|$, and we use a batch size of 1024 in this table. Here, TFLOPS is the quantity of computation to perform.

DNN architecture	typical use-case	data_size $\|D\|$	weight_size $\|W\|$	data/weight ratio	Forward+Backward TFLOPS/batch
NiN [30]	computer vision	5800MB	30MB	195	6.7TF
AlexNet [26]	computer vision	1680MB	249MB	10.2	7.0TF
GoogLeNet [39]	computer vision	19100MB	54MB	358	9.7TF
VGG-19 [37]	computer vision	42700MB	575MB	71.7	120TF
MSFT-Speech [36]	speech recognition	74MB	151MB	0.50	0.00015TF

Parallelism strategies

- Each worker (GPU) gets a subset of each batch (32 x 32 = 1024).

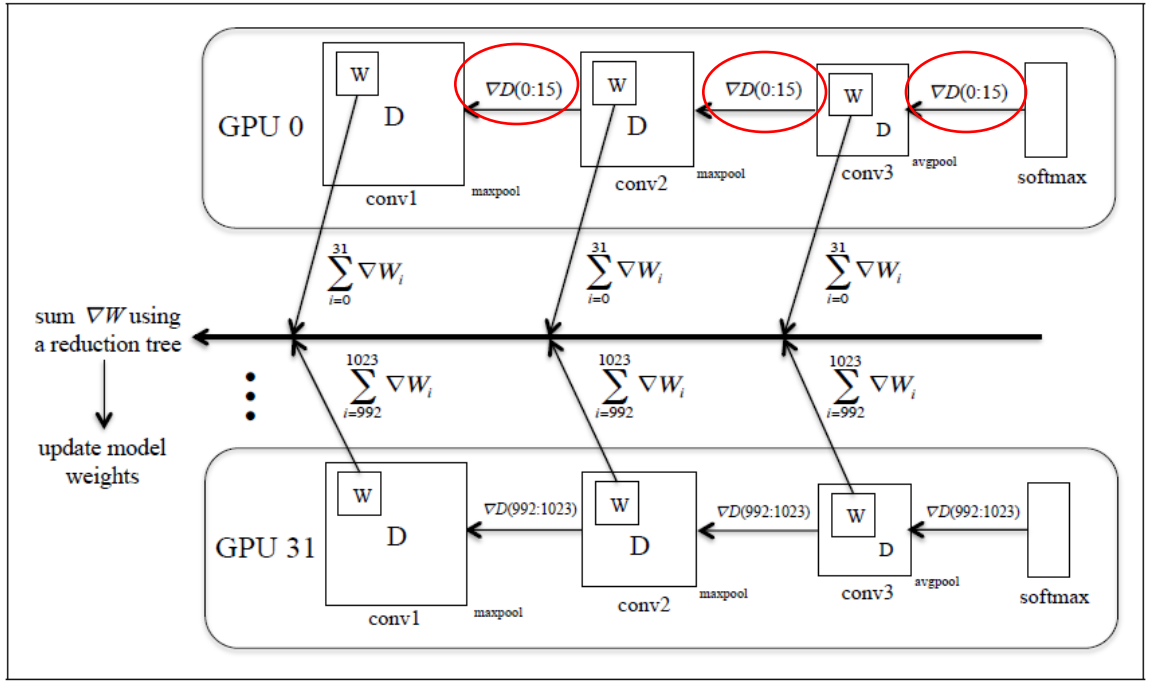


Figure 1. **Data parallel DNN training in FireCaffe:** Each worker (GPU) gets a subset of each batch.

Misprinting 1

Choosing DNN architectures to accelerate

DNN models with more parameters would have higher classification accuracy?

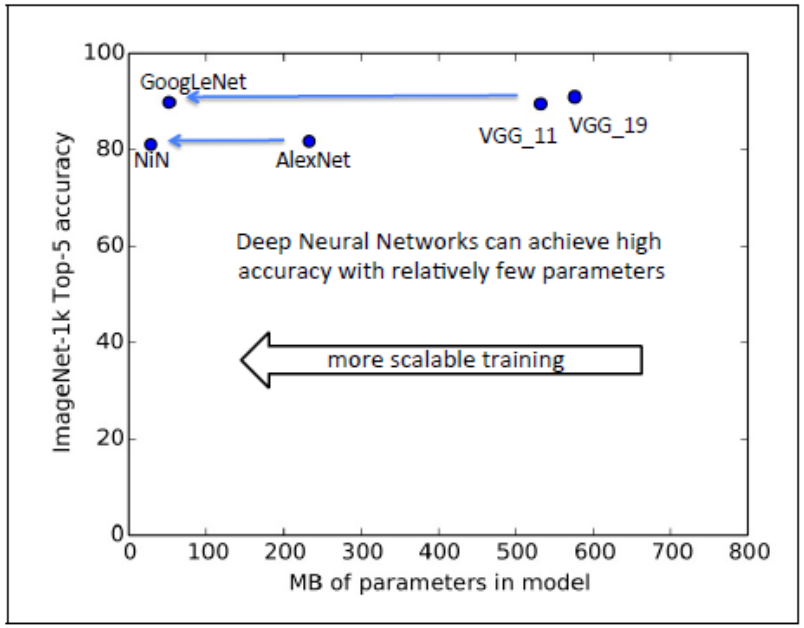
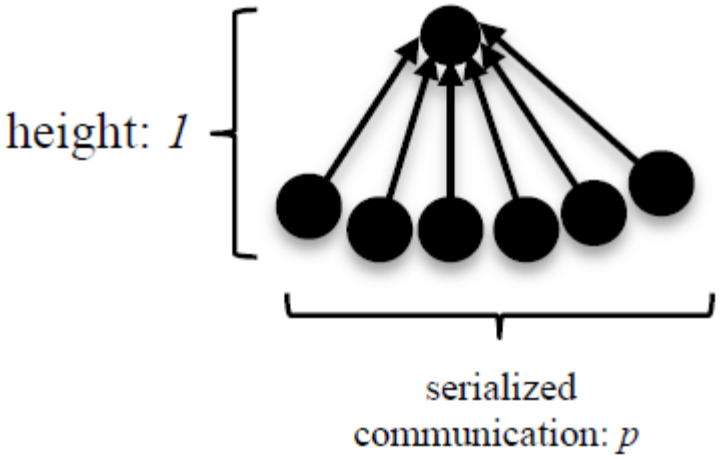


Figure 2. Deep neural network architectures with more parameters do not necessarily deliver higher accuracy.

Methodology of data parallel training

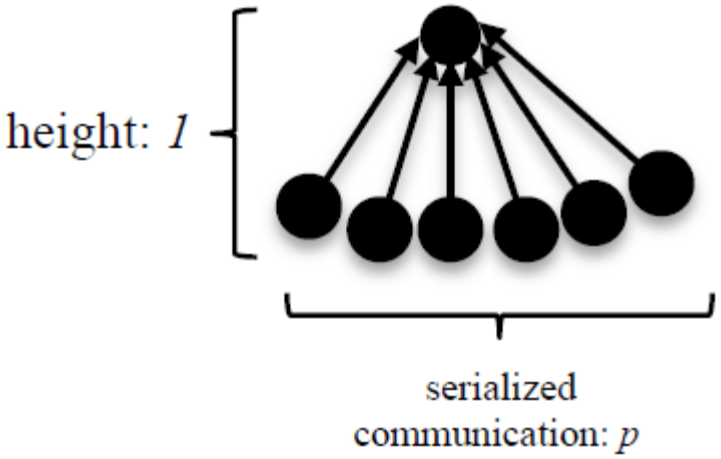
- No communication among GPU workers in the *forward pass*.
- Distributing the *backward pass* over a compute cluster, each GPU worker computes a sum of the weight gradients ($\sum \nabla W$) for its subset of the batch.
- Summing the weight gradients for a batch across GPUs.
- Results are as same as using a single GPU.
- Parameter servers or Reduction trees.

Methodology 1: Parameter server



1. Appointing one node as a parameter server.
2. Rest nodes are assigned a subset of the batch to perform forward pass and backward pass.
3. The parameter server computes the sum of the gradients.
4. The parameter server sends the summed gradients to the workers, and the workers apply these gradient updates to their local copies of the model.

Methodology 1: Parameter server

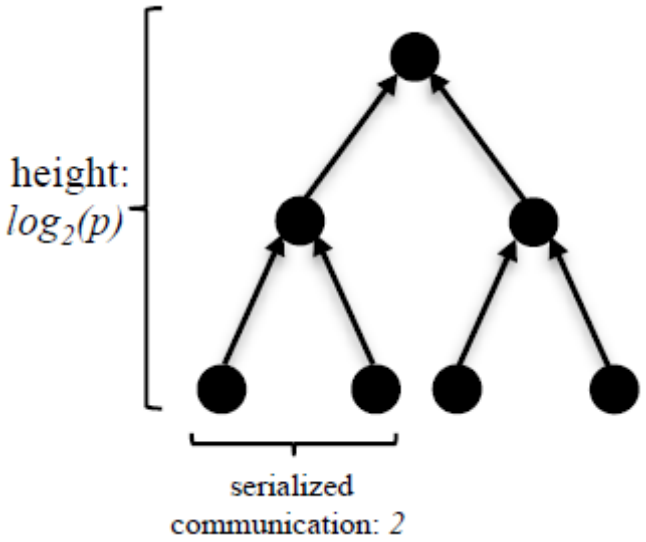


- Each GPU worker provides $|W| = |\nabla W|$ bytes weight gradients (Equation 1)
- p GPU workers.
- $|\nabla W|$ * p bytes of data(the parameter server).
- Send and receive data at a rate of BW bytes/s.

$$param\_server\_communication\_time = \frac{|\nabla W| * p}{BW}(sec) \quad (3)$$

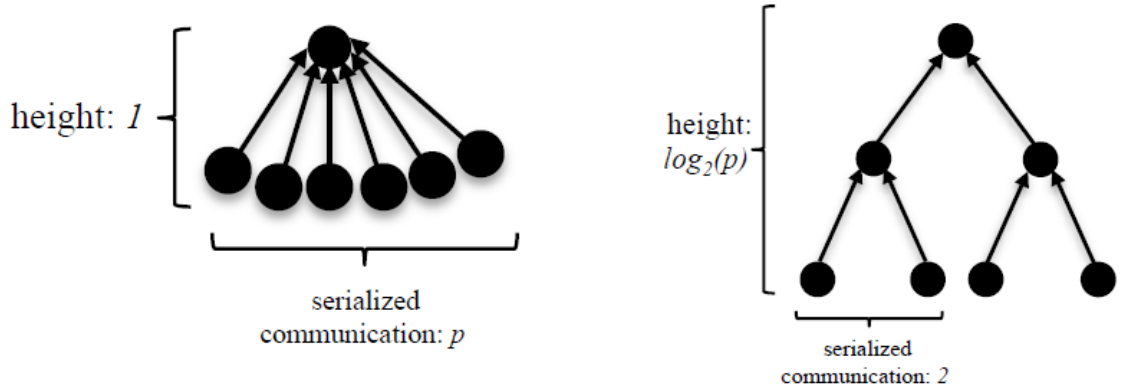
The parameter server's communication time scales linearly as increasing the number of GPU workers

Methodology 2: Reduction tree



- Allreduce: Reduce followed by an Bcast.
- Binomial reduction tree is easy to make the operation scales in the log of the number of workers p.

Difference between Parameter server and Reduction tree



(a) parameter server (b) reduction tree

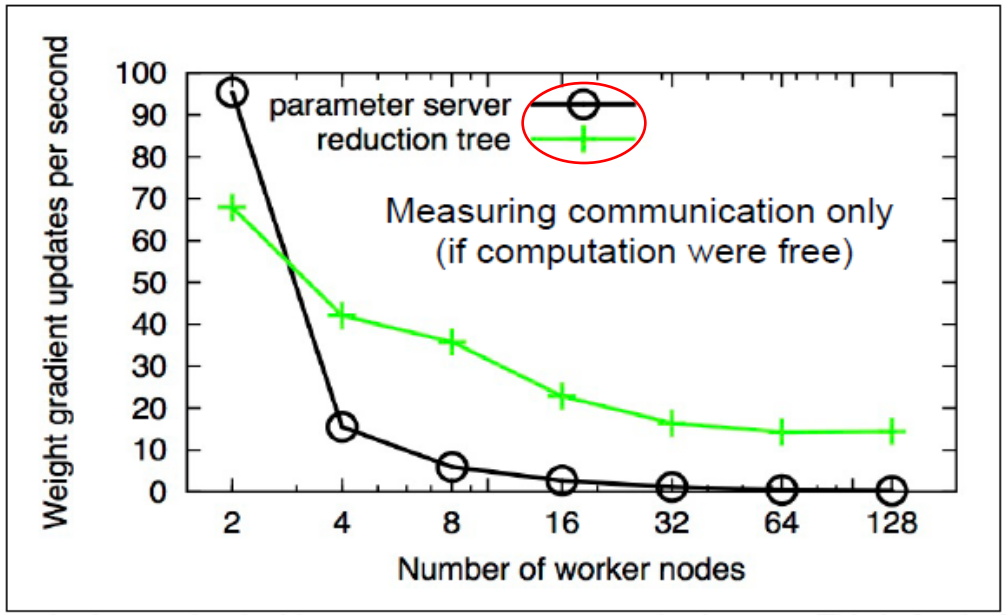
Figure 3. Illustrating how parameter servers and reduction trees communicate weight gradients. In this figure, we only show the summing-up of weight gradients. We distribute the weight gradient sums by going back down the tree.

Height: 1
branching factor: p

Height: log2(p)
branching factor: 2

$$reduction\_tree\_communication\_time = \frac{|\nabla W| * 2log_2(p)}{BW}(sec) \quad (4)$$

Difference between Parameter server and Reduction tree



Misprinting 2?

Figure 4. Comparing communication overhead with a parameter server vs. a reduction tree. This is for the Network-in-Network DNN architecture, so each GPU worker contributes 30MB of gradient updates.

$$param\_server\_communication\_time = \frac{|\nabla W| * p}{BW}(sec) \quad (3)$$

$$reduction\_tree\_communication\_time = \frac{|\nabla W| * 2log_2(p)}{BW}(sec) \quad (4)$$

Evaluation: Hardware and Data set

- ImageNet-1k
  - more than 1 million training images
  - each image is labeled as containing 1 of 1000 object categories
- GoogLeNet and Network-in-Network
- A GPU cluster with NVIDIA Kepler-based K20x GPU per server in the OLCF Titan supercomputer
- Reduction tree
- Data parallelism

Evaluation : Hyperparameter settings

- Hyperparameter settings have a big impact on the speed and accuracy produced in DNN training.
  - NiN: Caffe configuration files released by the NiN authors[6]
  - GoogLeNet: follow settings with [7][8][9]

- Address hyperparameter settings in the following sections.

[6] M. Lin, Q. Chen, and S. Yan. Network in network. arXiv:1312.4400
[7] Z. Wu, Y. Zhang, F. Yu, and J. Xiao. A gpu implementation of googlenet. http://vision.princeton.edu/pvt/GoogLeNet
[8] Guadarrama. BVLC googlenet. https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet
[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. JMLR

Evaluation : A single-server baseline

- Caffe on a single-server
- Speedups
- Accuracy

Result1: Midsized deep models

Table 2. Accelerating the training of midsized deep models on ImageNet-1k.

	Hardware	Net	Epochs	Batch size	Initial Learning Rate	Train time	Speedup	Top-1 Accuracy
Caffe [25]	1 NVIDIA K20	AlexNet [27]	100	256	0.01	6.0 days	1x	58.9%
Caffe	1 NVIDIA K20	NiN [30]	47	256	0.01	5.8 days	1x	58.9%
Google [26]	8 NVIDIA K20s (1 node)	AlexNet	100	varies	0.02	16 hours	7.7x	57.1%
FireCaffe (ours)	32 NVIDIA K20s (Titan supercomputer)	NiN	47	256	0.01	11 hours	13x	58.9%
FireCaffe-batch1024 (ours)	32 NVIDIA K20s (Titan supercomputer)	NiN	47	1024	0.04	6 hours	**23x**	58.6%

- Fixed number of epochs at 47.
- Increasing the batch size.
- Reduces the number of times we need to communicate weight gradients
- Reducing the overall training time

Result2: Ultra deep models

Table 3. Accelerating the training of ultra-deep, computationally intensive models on ImageNet-1k.

	Hardware	Net	Epochs	Batch size	Initial Learning Rate	Train time	Speedup	Top-1 Accuracy	Top-5 Accuracy
Caffe	1 NVIDIA K20	GoogLeNet [39]	64	32	0.01	21 days	1x	68.3%	88.7%
FireCaffe (ours)	32 NVIDIA K20s (Titan supercomputer)	GoogLeNet	72	1024	0.08	1.3 days	**16x**	68.3%	88.7%

- Larger batch sizes lead to less communication
- Learning rate is crucial in order to preserve high accuracy
- Learning rate (LR): {0.02, 0.04, 0.08, 0.16, and 0.32}
  - LR=0.16 and LR=0.32, didn't ever learn anything beyond random-chance accuracy on the test set.
  - LR=0.02 , 66.1% top-1, and LR=0.04 produced 67.2%.

Conclusions

- This paper focus on the problem of accelerating DNN training, and our work has culminated in the FireCaffe distributed DNN training system with three key pillars.
  - Network hardware that achieves high bandwidth between GPU servers.
  - Selecting a communication algorithm (reduction trees)
  - Optionally increase the batch size and identify hyperparameters
- After eveluations, this paper found the approximately balance of communication at the 32-GPU scale.
  - to achieve a near-linear speedup for a number of leading deep neural network architectures.
  - In particular, we have achieved 23x speedup in NiN training, and a 16x speedup on GoogLeNet training on a 32 GPU cluster.

My Impression

- Deficiencies
  - We need more details of evaluation results such as overhead of communication and computation with different cluster size and batch size.
  - Did not try the model parallelism.

- TO DO
  - Firecaffe is implemented on a GPUs cluster, in which every node has only one GPU, Instead TSUBAME-KFC has 42 nodes, and each node has K80 GPU x 2. I would like to test it on TSUBAME-KFC by using 2 GPUs within a single node firstly like Krizhevsky did At Google. Then, reduction tree parallelization within a GPU cluster with adjusting hyperparameters such as Epochs, batch size, learning rate and so on.