Poseidon: A System Architecture for Efficient GPU-based Deep Learning on Multiple Machines

2016 USENIX Annual Technical Conference
JUNE 22–24, 2016
DENVER, CO
www.usenix.org/atc16

Hao Zhang, Zhiting Hu, Jinliang Wei, Pengtao Xie, Gunhee Kim, Qirong Ho, Eric Xing
{hao, zhitingh, jinlianw, pengtaox, gunhee, epxing}@cs.cmu.edu

Introduction

We propose a scalable open-source framework for large-scale distributed deep learning on GPU clusters. We build the framework upon the Caffe CNN libraries and the Petuum distributed ML framework as a starting point, but goes further by implementing three key contributions for efficient CNN training on clusters of GPU-equipped machines: (i) a three-level hybrid architecture to support both CPU-only clusters as well as GPU-equipped clusters, (ii) a distributed wait-free backpropagation (DWBP) algorithm to improve GPU utilization and to balance communication, and (iii) a dedicated structure-aware communication protocol (SACP) to minimize communication overheads. We empirically show that our framework converges to the same objective value as a single machine, and achieves state-of-art training speedup across multiple models and well-established datasets, using a commodity GPU cluster of 8 nodes.

Background: Iterative-Convergent Algorithms

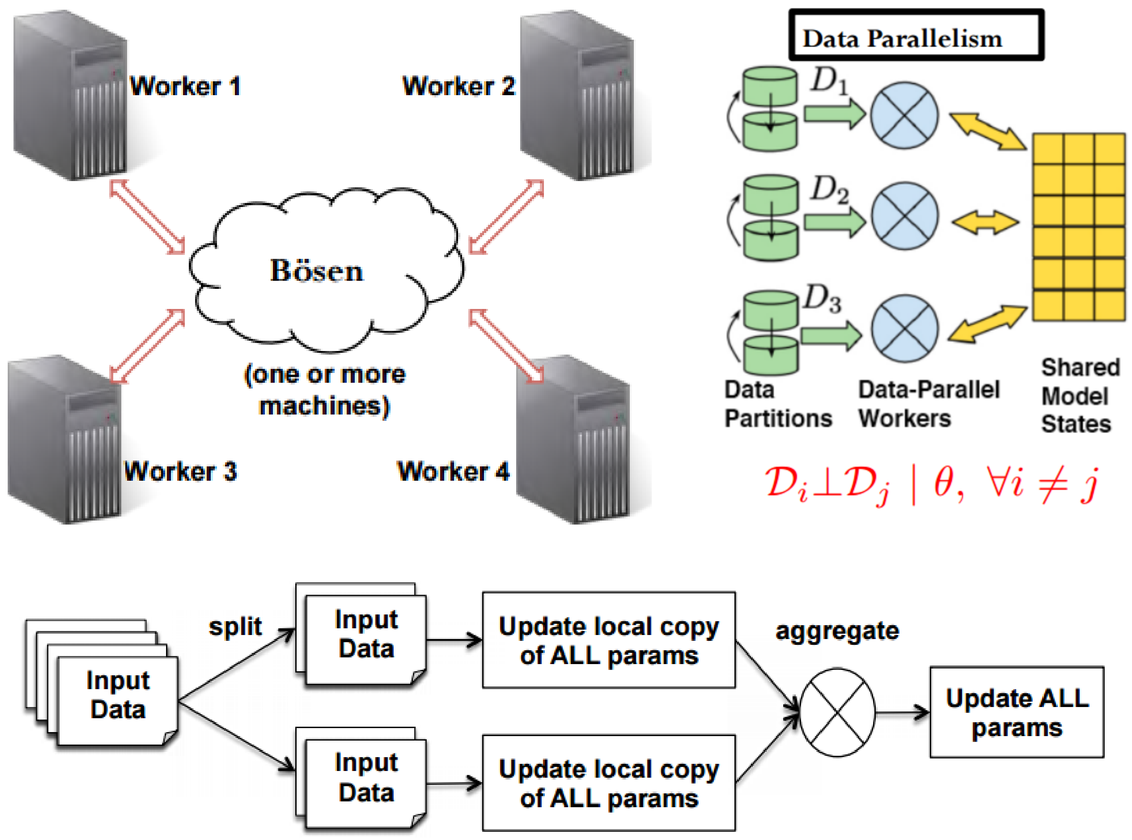
Iterative-Convergent Algorithm in General

The *iterative-convergent* algorithm can be represented as follows.

$$A^{(t)} = F\left(A^{(t-1)}, \Delta_\ell\left(A^{(t-1)}, D\right)\right)$$

In large-scale machine learning, both data D and model A can be very large.

$$A^{(t)} = F\left(A^{(t-1)}, \sum_{p=1}^{P} \Delta_\ell\left(A^{(t-1)}, D_p\right)\right)$$



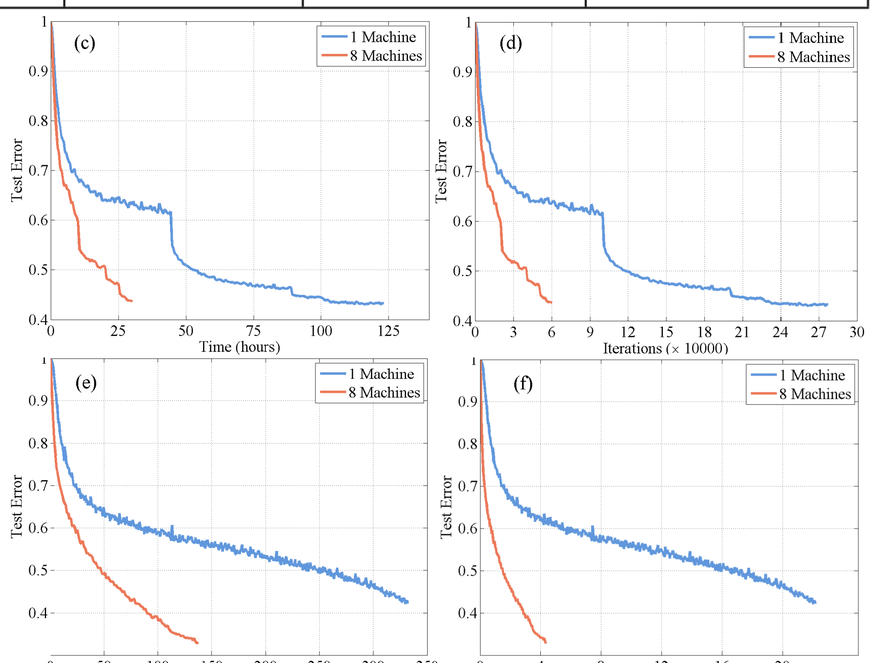
Experiments

Accelerate the Training of Modern CNNs

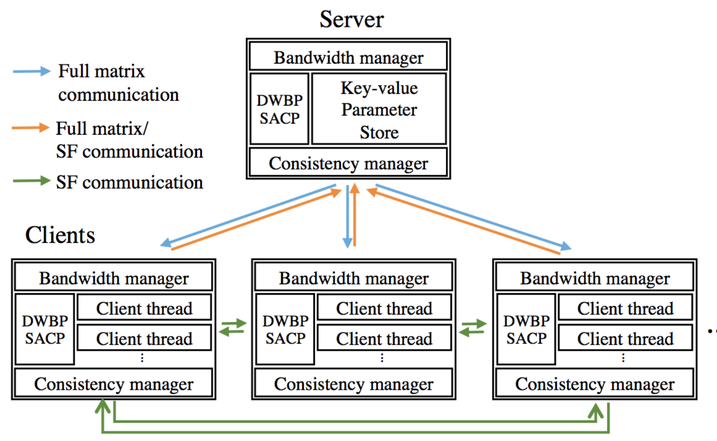
Dataset	# of Images	Size of images	# of categories
ILSVRC2012	1.3M	$256 \times 256 \times 3$	1000
ImageNet22K	15M	$256 \times 256 \times 3$	21841

AlexNet Training
4.5x speedup 8 nodes
56.5% top-1 accuracy

GoogLeNet Training
4x speedup 8 nodes
67.1% top-1 accuracy



System Architecture

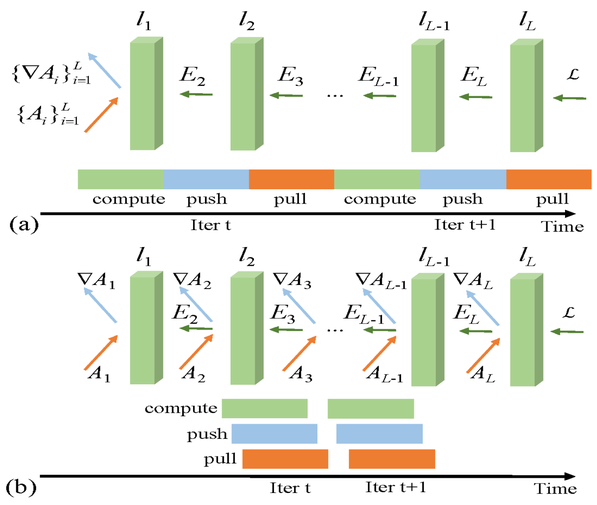


Algorithm 1: CNN training with data-parallelism on Poseidon

Slave nodes:
1. Partition training data D equally into $\{D_i\}_{i=1}^{P}$ and distribute them to all P nodes.
2. Replicate the initial model parameters A to every worker thread p as A_p.
3. **for** $t = 1$ **to** T **do**
4. **foreach** *worker thread* $p \in \{1, 2, \cdots, P\}$ **do**
5. Take a batch of training data D_p' from D_p.
6. Perform forward pass.
7. Perform backpropagation pass following the DWBP algorithm.
8. Update local synchronization states to the consistency manager.

Master node:
1. **for** $t = 1$ **to** T **do**
2. Collect gradients that are sent by worker nodes.
3. Updates the part of model parameters for which a corresponding gradient is received.
4. Push updated model parameters to worker nodes according to the consistency manager.

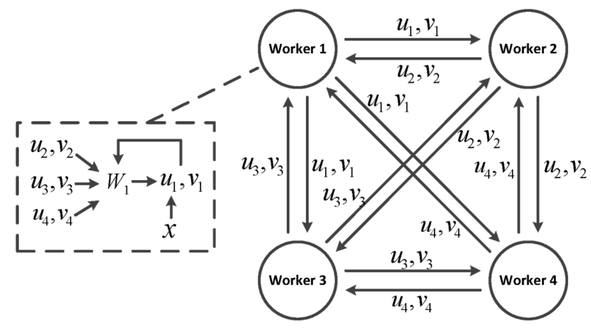
Distributed Wait-free Backpropagation (DWBP)



Algorithm 2: The Distributed Wait-free Backpropagation (DWBP) Algorithm

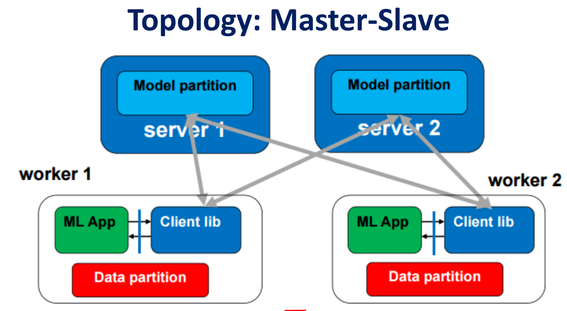
At iteration t on worker p:
Input: Loss ℓ.
1. **for** $i = L$ **to** 1 **do**
2. **if** $i == L$ **then**
3. Compute gradients $\nabla A_i = \frac{\partial \ell}{\partial A_i}$ using ℓ.
4. **else**
5. Receive error message E_{i+1} from layer l_{i+1}.
6. Compute gradients $\nabla A_i = \frac{\partial \ell}{\partial A_i}$ using E_{i+1}.
7. **if** $i \neq 1$ **then**
8. Compute error message E_i and pass to layer l_{i-1}.
9. Communicate: push out ∇A_i and pull in updated A_i following the SACP protocol;

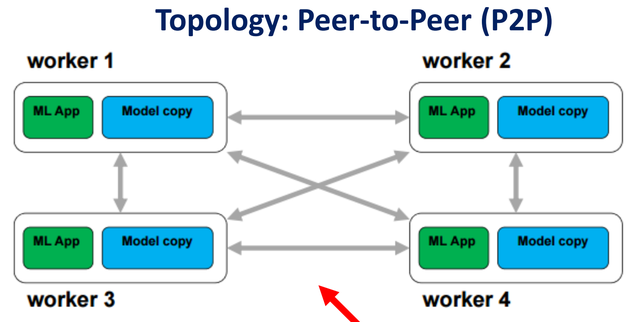
Structure-Aware Communication Protocol (SACP)

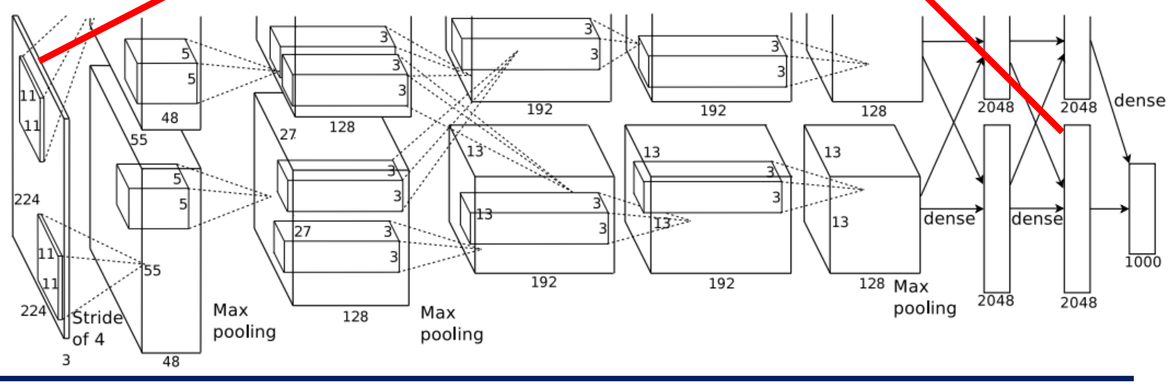


FC Layers in AlexNet = 4096 * 4096 (17M)
Use Sufficient Factors(SF) to commutate:
1. Decouple into two 4096 vectors: u and v
2. Transmit two vectors
3. Reconstruct the gradient matrix

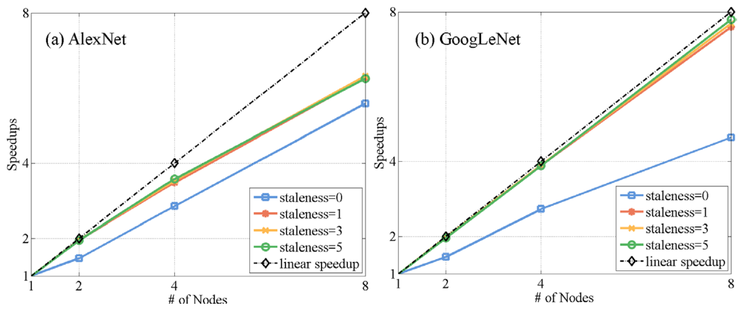
Because the rank of the gradient matrix is 1!







The speedups on throughput with different values of staleness, (a) Training AlexNet with batch size 256, and (b) Training GoogLeNet with batch size 32.



Training AlexNet and GoogLeNet with different number of GPU nodes and settings: (a) AlexNet with batch size 256 ; (b) GoogLeNet with batch size 32 . Compared to single machine Caffe