Parallelization of Neural Network Training using Model Averaging and Butterfly Mixing

Hang Su[1,2], Haoyu Chen[2]
[1] International Computer Science Institute, CA, USA [2] Dept. of EECS, UC Berkeley, CA, USA

Goal

- Distributed neural network training using multiple GPUs.
- Model average for distributed network training.
- Apply different network communication strategies (all-reduce, butterfly and ring).

Background

Condition

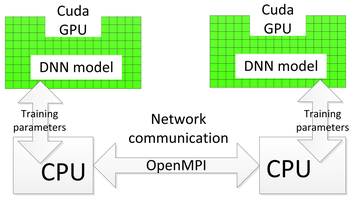
- Large scale continuous time automatic speech recognition task.
- Deep Neural Network with fully connected layers and thousands of hidden nodes.
- Multiple GPUs available on different computing nodes.
- Data communication among the nodes are bottleneck for distributed training.

Approaches

- Data parallelism – train neural networks locally using partitioned data (GPU based).
- Average model parameters among computing nodes every few minibatches (OpenMPI based).
- Apply butterfly / ring mixing network communication strategy to reduce data communication.
- Special training algorithm (natural gradient SGD) for better covergence rate.

Setup

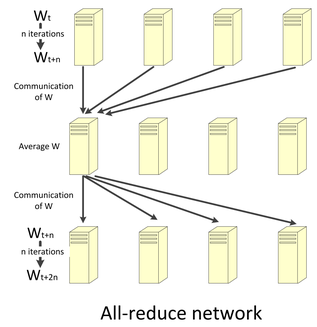
- OpenMPI for data communicate between nodes.
- GPU for local neural network training



Parallelization Strategy

All-reduce network

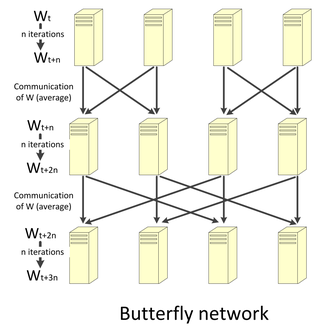
After every n local updates, all the nodes reduce their parameter together (average), and broadcast the parameter back to all the nodes. This requires high communication bandwidth.



All-reduce network

Butterfly-mixing

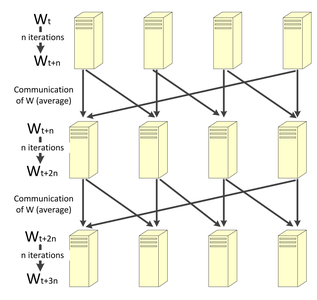
In Butterfly-mixing strategy, one node only communicates with one other node, averaging the recieved model with its own model. After $log(nodeSize)$ communication, the information is fully communicated.



Butterfly network

Ring network

In ring network, each node sends its parameter to its right neighbor and receive parameters from its left neighbor. After $nodeSize$ communication, the information is fully communicated. This method benefits from the locality of computing nodes.



Ring network

Natural Gradient SGD

- Take steps along the gradient of a Riemannian parameter surface.
- Use inverse fisher matrix E_t as learning-rate matrix.
- α denotes original learning rate, g_t denotes plain gradient. Update parameters using:

$$\theta_{t+1} = \theta_t + \alpha_t E_t g_t \quad (1)$$

Experiments

Convergence Status

Table 1: Convergence status on CV set

Nodes		1	2	4	8	16
Allreduce	Obj (loss)	1.43	1.57	1.45	1.45	1.48
	Frame Acc(%)	59.7	59.3	60.2	59.9	58.5
Butterfly	Obj (loss)	–	–	1.47	1.52	1.70
	Frame Acc(%)	–	–	60.0	58.4	54.7
Ring	Obj (loss)	–	–	1.47	1.62	1.96
	Frame Acc(%)	–	–	59.8	56.2	50.7

Scaling factor

Table 2: Speedup and scaling factor

Nodes		1	2	4	8	16
Allreduce	Time	6.59	3.39	1.90	1.01	0.700
	speedup	–	1.94	3.47	6.52	9.41
Butterfly	Time	–	–	1.83	1.00	0.582
	speedup	–	–	3.60	6.59	11.32
Ring	Time	–	–	1.84	1.01	0.517
	speedup	–	–	3.58	6.52	12.75

Recognition Results

Table 3: Word error rate on different setup

Nodes	1	2	4	8	16
allreduce	18.4	17.9	18.3	18.3	18.7
butterfly	–	–	18.5	18.7	20.6
ring	–	–	18.5	19.6	22.1

Conclusion

- Neural network training can be efficiently speed up by using model averaging techniques.
- Natural gradient method contributes to the convergence rate in model averaging setup.
- Butterfly-mixing and ring reduction can reduce communication time in a further step.
- Training using Butterfly-mixing and ring reduction does not converge well as the number of jobs goes up to 16.