Detailed Performance Analysis of Distributed Tensorflow on a GPU Cluster using Deep Learning Algorithms

Abid Malik, *Yuewei Lin, Shinjae Yoo, Nathaniel Wang, and Michael Lu*

Computer Science and Mathematics Department

Computational Science Initiative

NYSDS 2018

BROOKHAVEN NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

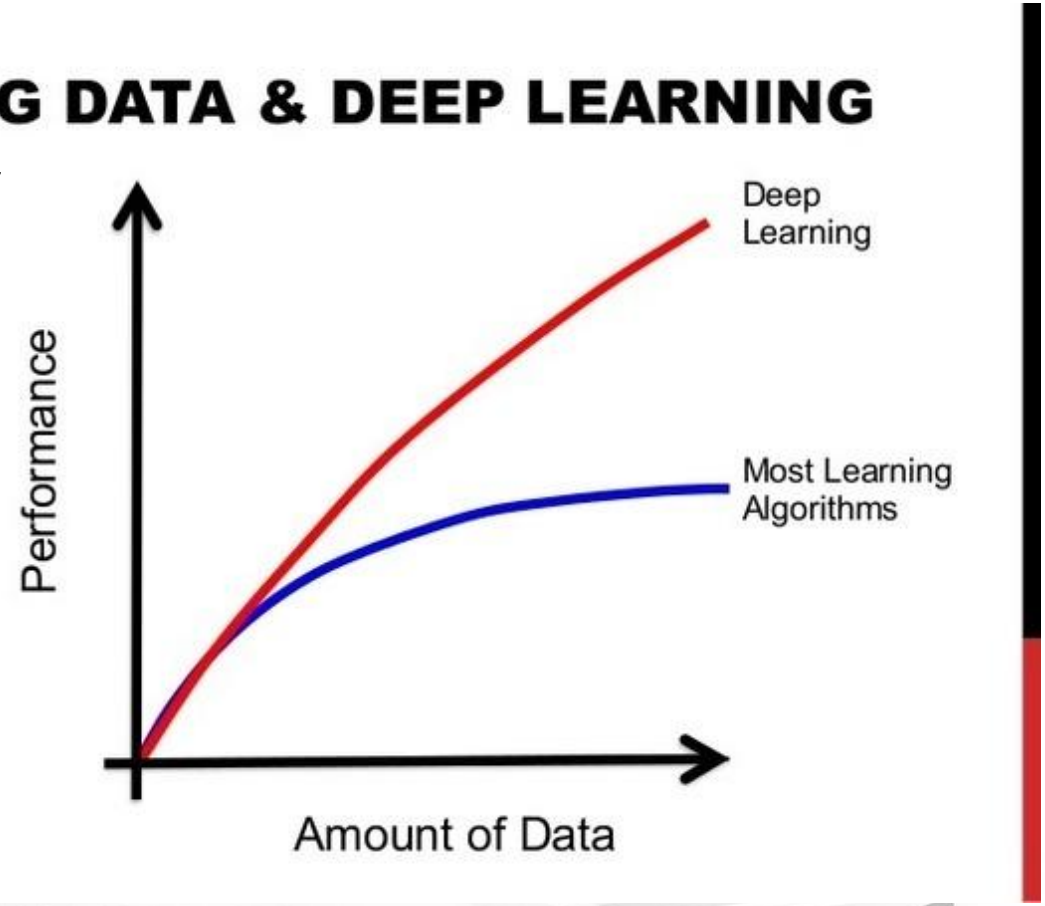
Outline

- Introduction
- Distributed Deep Neural Networks
- Horovod Framework
- Detailed Performance Analysis
- Conclusion and Future Work

Introduction

Where there is a large data, there is a need for deep learning to process the information!

We find Big Data in every field. Therefore, we find great demand of deep learning in every area

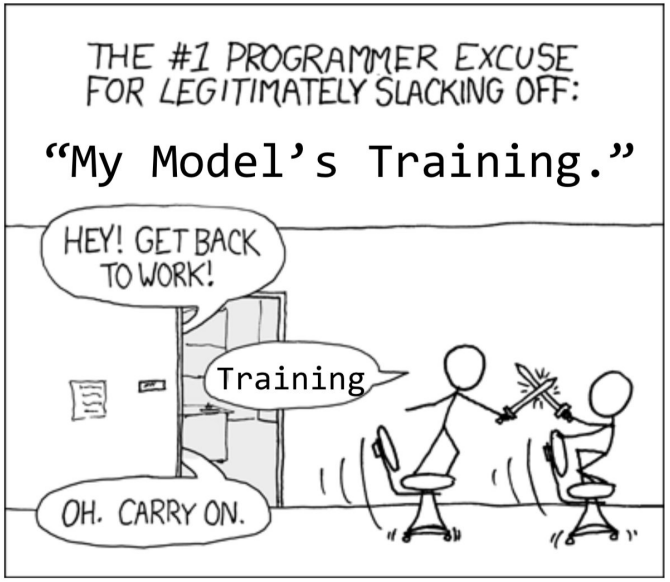


Why we need High Performance Computing (HPC) for Deep Learning?

Jeffrey Deans from the Google Tensorflow Team

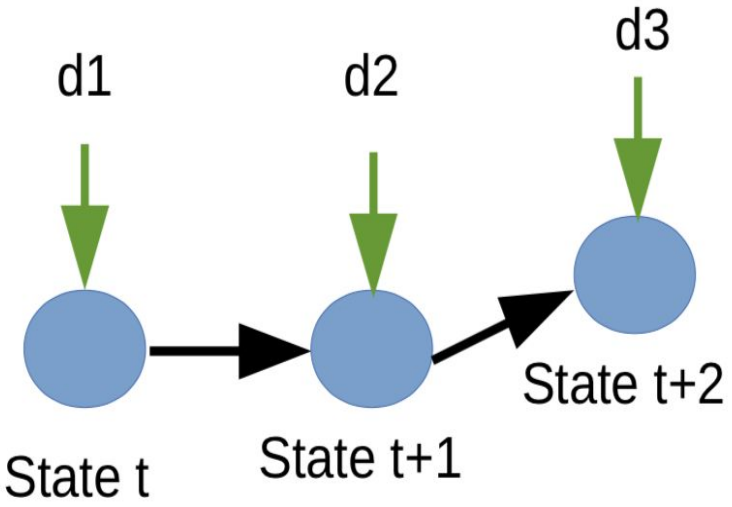
- > 1 month
  - Don't even try
- 1-4 weeks
  - High value experiments only
  - Progress stalls
- 1-4 days
  - Tolerable
  - Interactively replaced by running many experiments in parallel
- Minutes, Hours
  - Interactive research! Instant gratification! ☺
- Online learning, training time is crucial

Shorter "training time" is important, and we need HPC for this!



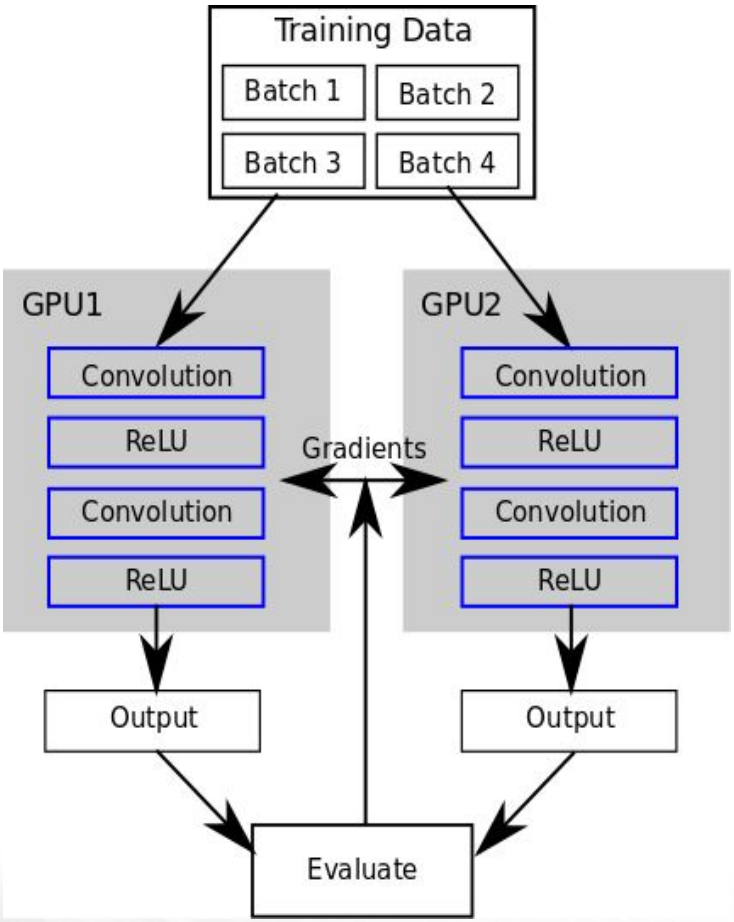
Parallelizing Deep Learning

- **Stochastic gradient descent** (often shortened to **SGD**), also known as **incremental** gradient descent, is an iterative method for optimizing a differentiable objective function
- Parallelizing SGD is very hard. It is inherently sequential algorithm
  1. Start at some state **t** (point in a billion dimensional space)
  2. Introduce **t** to data batch **d1**
  3. Compute an update (based on the objective function)
  4. Apply the update → **t+1**

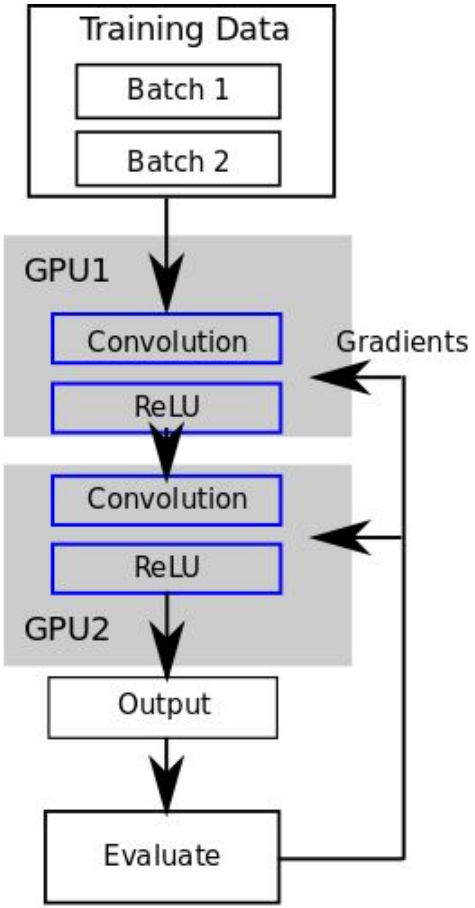


Parallelizing Deep Learning

Data Parallelism

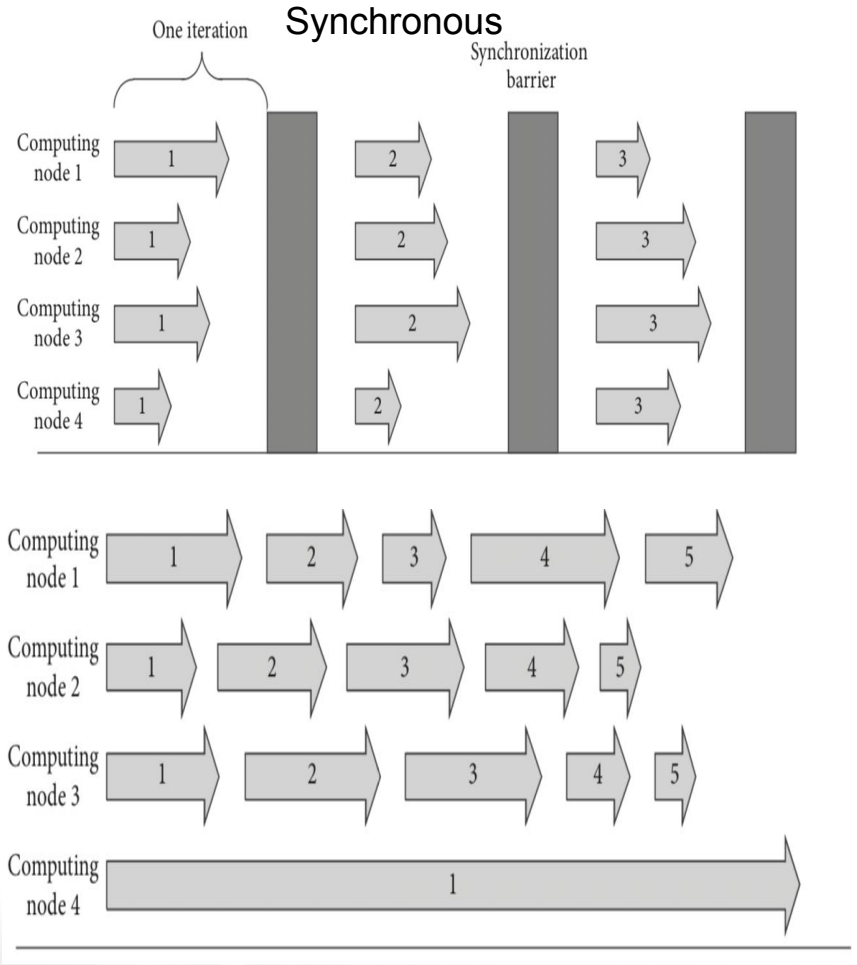


Model Parallelism

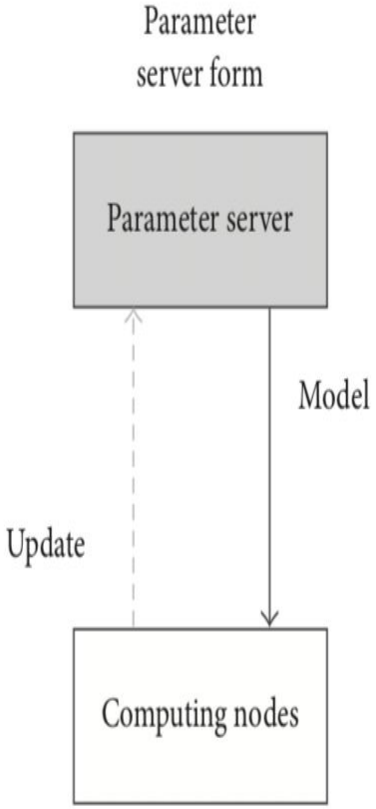


Parallelizing Deep Learning

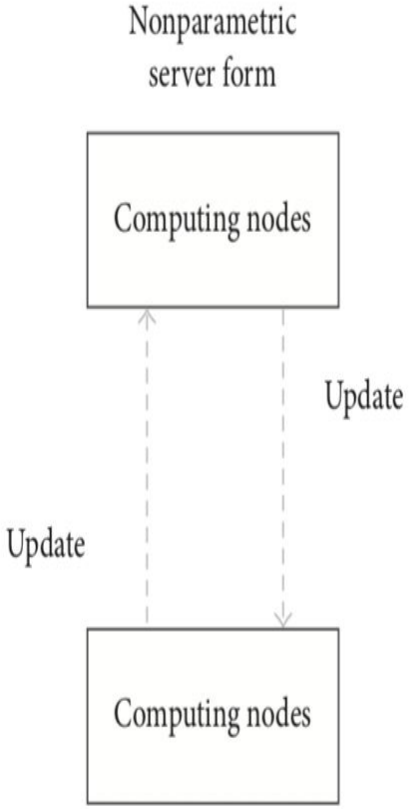
Synchronous



Asynchronous



Centralized



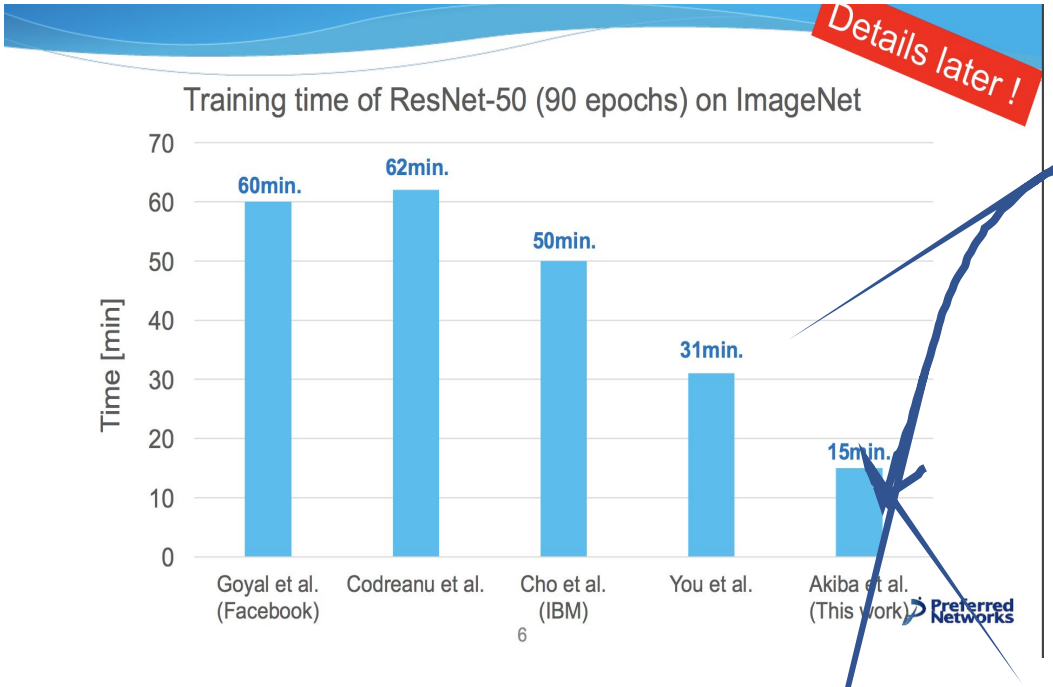
Non-Centralized

Scalability Performance

- Three core performance parameters:
  - **Throughput** : data points/sec
  - **Latency** : completion time for one epoch
  - **Accuracy**: ability to classify unseen cases

- Need to improve the performance by:
  - Improving the performance on a node (needs lot of compiler optimizations)– internal parallelism
  - Improving the time between the two states (Input/output and communication is a big challenge)— external parallelism
  - Using large batch size

Noticeable Work in this Area!



Around 75% accuracy

The latest work

ImageNet Training in Minutes:Yang You (UC Berkeley), Zhao Zhang (TACC), Cho-Jui Hsieh (UC Davis), and James Demmel and Kurt Keutzer (UC Berkeley) - ICPP 2018

Extremely Large Mini-batch SGD: Training ResNet-50 on ImageNet in 15 Minutes Takuya Akiba, Shuji Suzuki, Keisuke Fukuda: 2017

Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes: Xianyan Jia et al. May, 2018

Writing Distributed Deep Learning Algorithms

- Challenges:
  - Deep learning framework or training library must support inter-node and intra-node communication
  - Tensorflow, MxNeT, PyTorch, Caffe, Chainer
  - These frameworks come with poorly understood overheads associated with communication and data management
  - The user must modify the code to take advantage of inter-node communication. The changes to code can be minimal to significant depending on the user's expertise in the distributed systems

Writing Distributed Deep Learning Algorithms

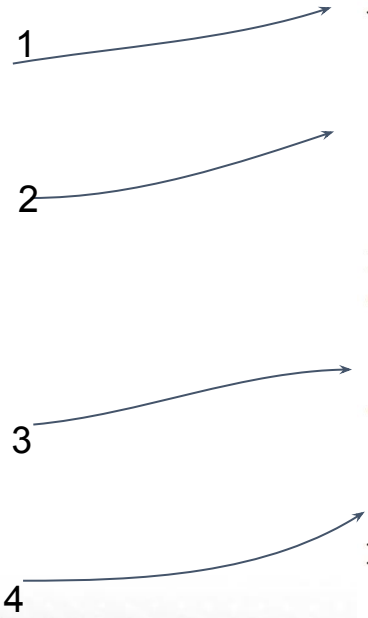
- Tensorflow supports Parameter Server model
- One needs to change 10% of your code to make Tensorflow code workable across the nodes
- One needs to tune the code to make it scalable over large number of nodes
- This requires an expert level knowledge of parallel programming models and distributed systems

Horovod Framework from Uber Inc.

- From Uber AI Engineering Team in 2017
  - Deep learning for self driving cars
  - Fraud detection
  - Optimal route prediction
- Open-source available at GitHub
- **Provides easy and fast way to do distributed machine learning using Tensorflow, PyTourch, and Keras**
- Adopted by **Cray, IBM,** and **Microsoft** for their distributed machine learning frameworks

Horovod Framework

- One needs to make few changes to transfer single-GPU programs to distributed GPU programs:



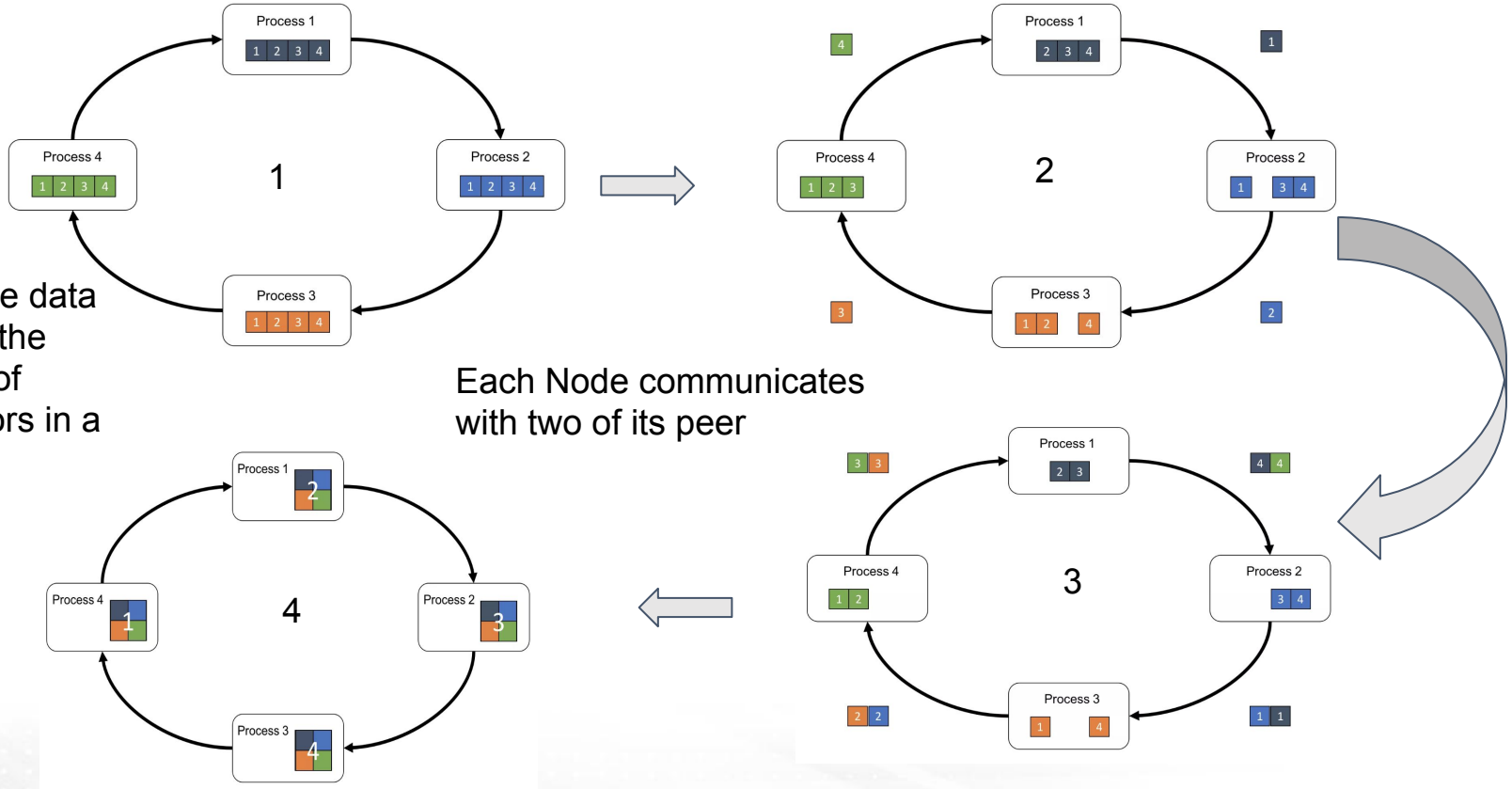

```python
import tensorflow as tf
import horovod.tensorflow as hvd

# Initialize Horovod
hvd.init()

# Pin GPU to be used to process local rank (one GPU per process)
config = tf.ConfigProto()
config.gpu_options.visible_device_list = str(hvd.local_rank())

# Build model...
loss = ...
opt = tf.train.AdagradOptimizer(0.01)

# Add Horovod Distributed Optimizer
opt = hvd.DistributedOptimizer(opt)

# Add hook to broadcast variables from rank 0 to all other processes
# during initialization.
hooks = [hvd.BroadcastGlobalVariablesHook(0)]

# Make training operation
train_op = opt.minimize(loss)
```

Ring All_Reduce

"Bandwidth Optimal All-reduce Algorithms for Cluster of Workstations" by Patarasuk and Yuan, 2009

Divide the data equal to the number of processors in a ring

Each Node communicates with two of its peer



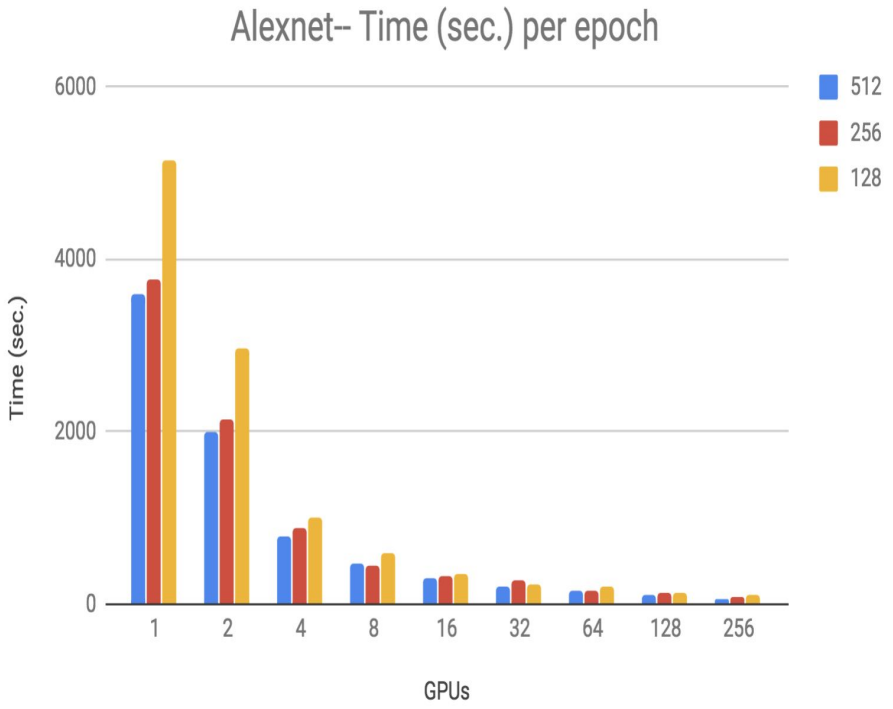
Ring All_Reduce

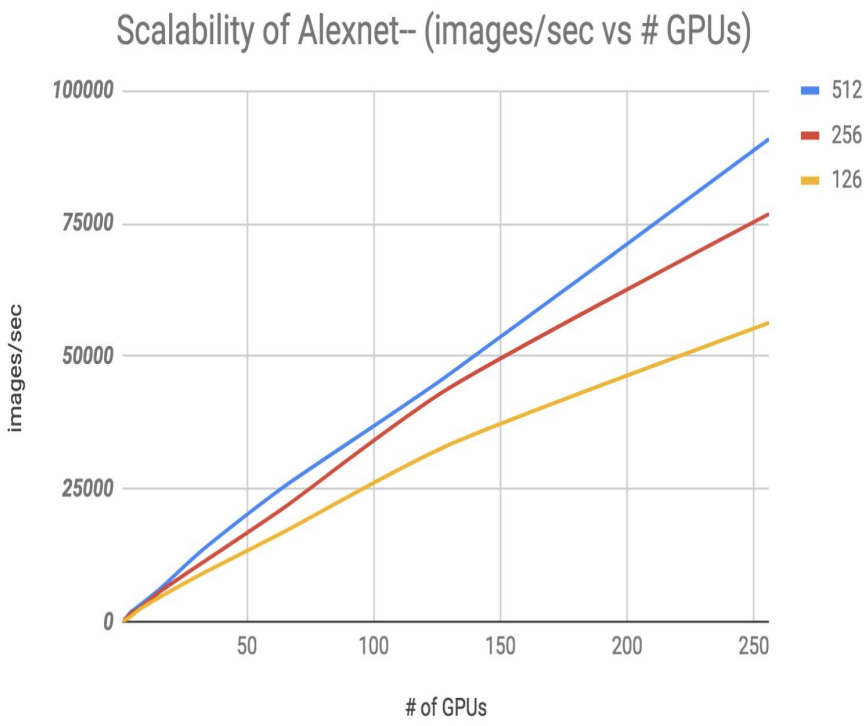
- The algorithm is bandwidth-optimal
- Each of **N** nodes communicates with two of its peers **2*(N-1)** times. During this communication, a node sends and receives chunks of the data buffer
- In the first **N-1** iterations, received values are added to the values in the node's buffer. In the second **N-1** iterations, received values replace the values held in the node's buffer
- The ring-allreduce algorithm allows worker nodes to average gradients and disperse them to all nodes without the need for a parameter server
- Baidu implemented it using MPI (the source code is available on Github)
- Horovod uses **NCCL** library for ring all_reduce implementation

Experimentation

- Detailed performance analysis of the Horovod Framework
- We used **AlexNet**, **GoogleNet**, and **ResNet50** implemented in Tensorflow
- We used synthetic data
- We used the ImageNet data
    - 1.2 million images
- We used batch sizes: 64,128, 256, and 512
- We used K20, K40 GPUs on HPC1 cluster at BNL
- We used Nvidia K80 and P100 GPUs on the Institutional Cluster at BNL
- 124 worker nodes (about 400 K80 GPUs)
- InfiniBand EDR connectivity

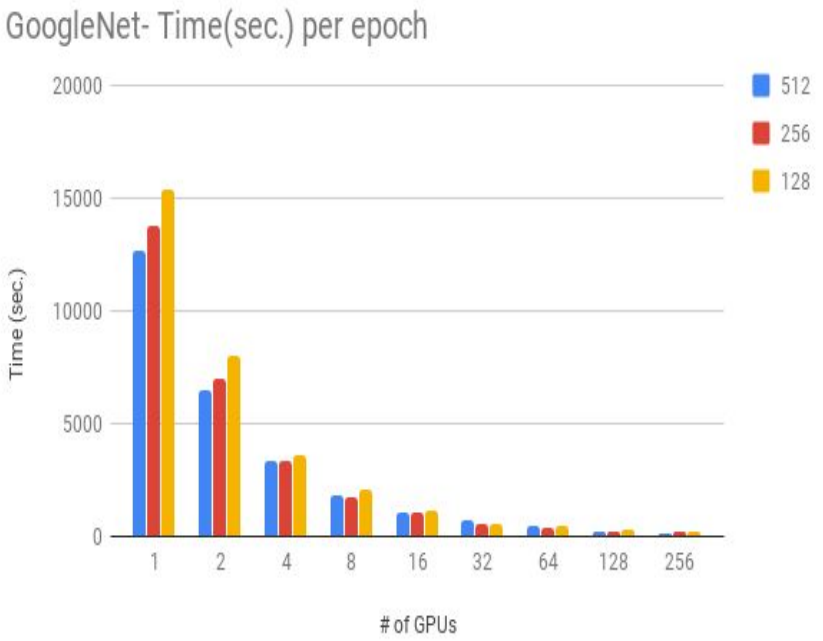
AlexNet using Horovod

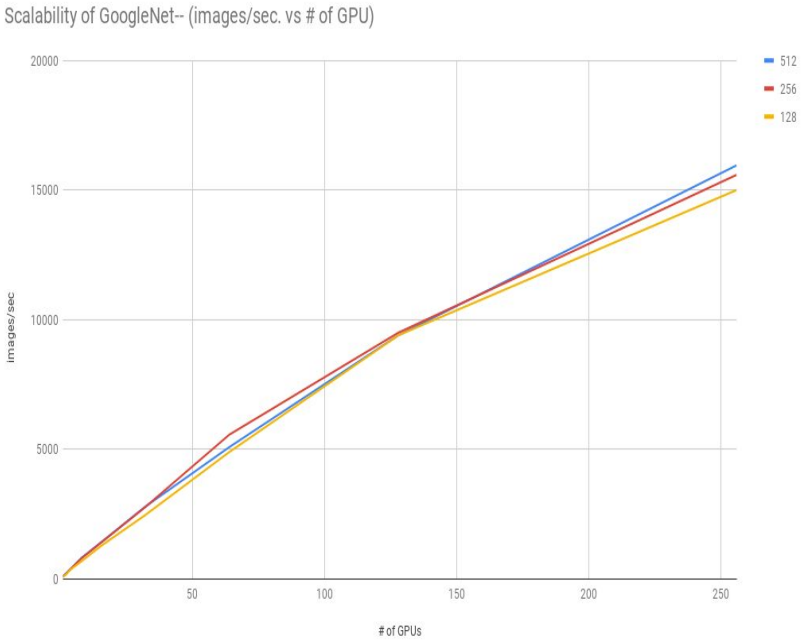




- Throughput performance and latency performance is almost linear
- larger batch size shows the best result

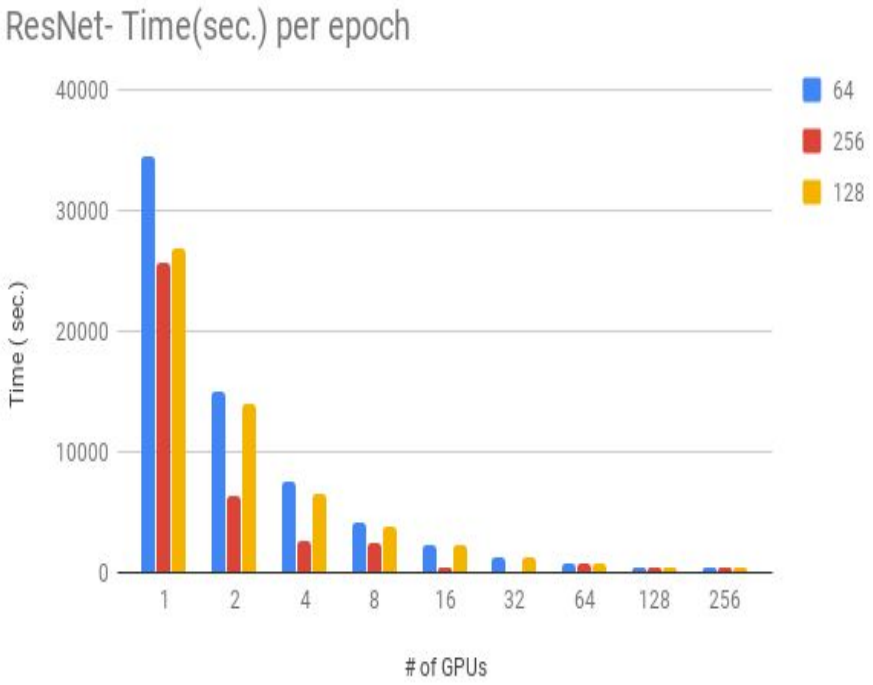
GoogleNet using Horovod





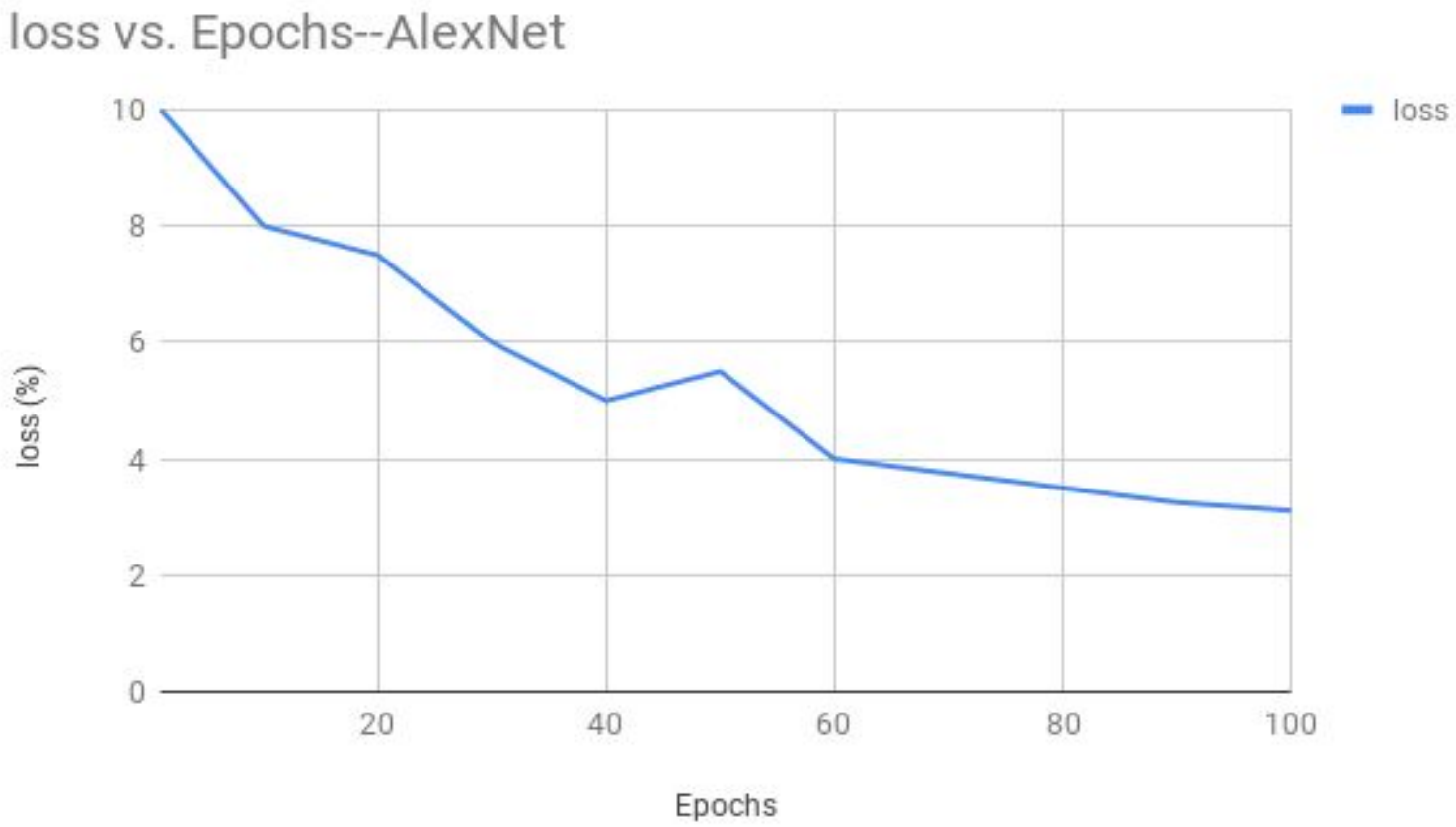
- Throughput performance and latency performance is almost linear
- All batch size shows same performance

ResNet50 using Horovod



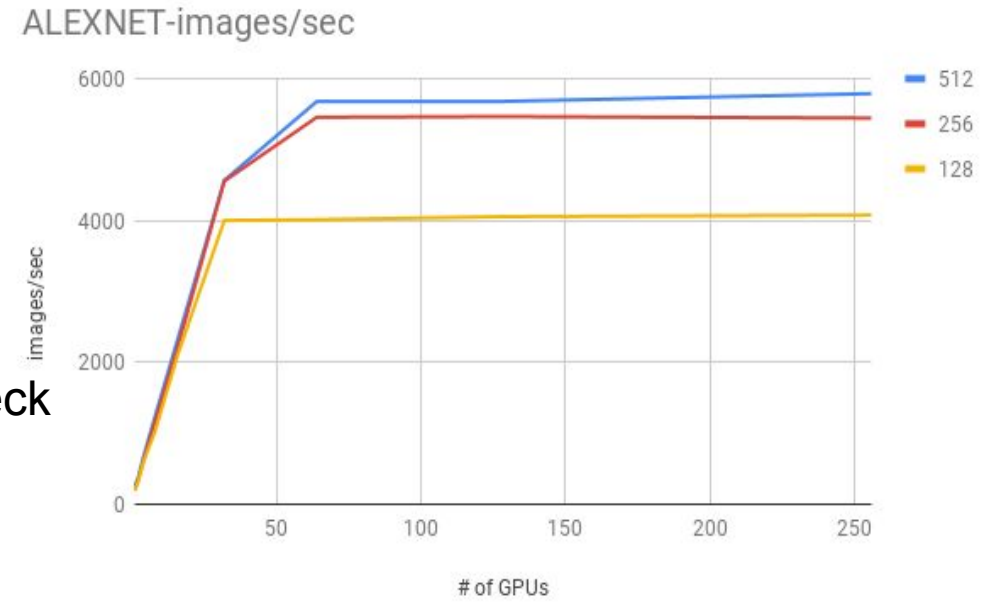
- Throughput performance and latency performance is almost linear
- All batch size shows same performance

Rate of Convergence using Horovod

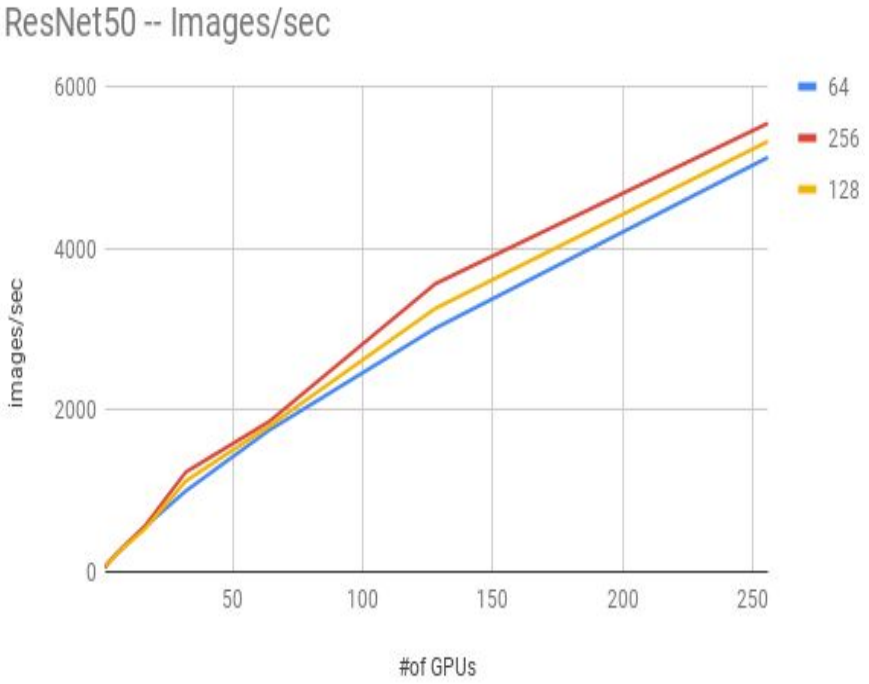


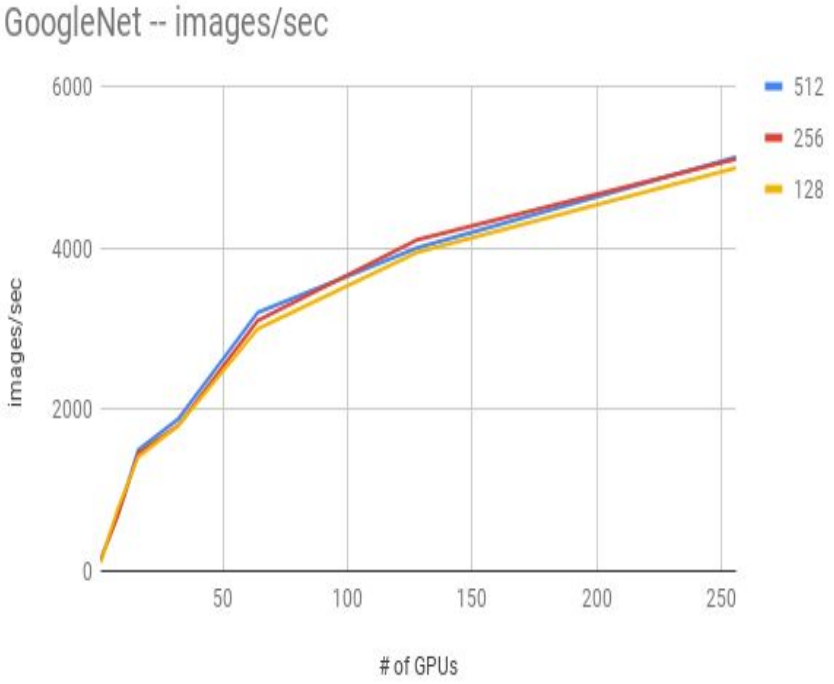
Using Parameter Server Tensorflow APIs

- Single Master Server which takes care of the parameter averaging (gradient update)
- The scalability in poor
- AlexNet has large number of parameters
- Communication is the main bottleneck here
- Interesting to do a detailed performance analysis
- Performance tool for distributed machine learning is not matured yet
- Working with TAU performance tool



Using Parameter Server Tensorflow APIs



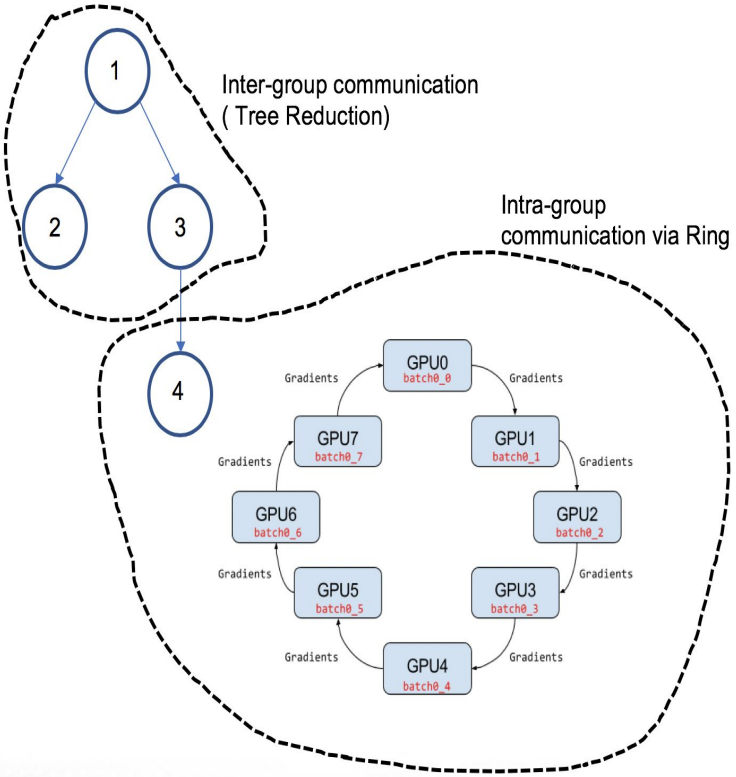


Conclusion

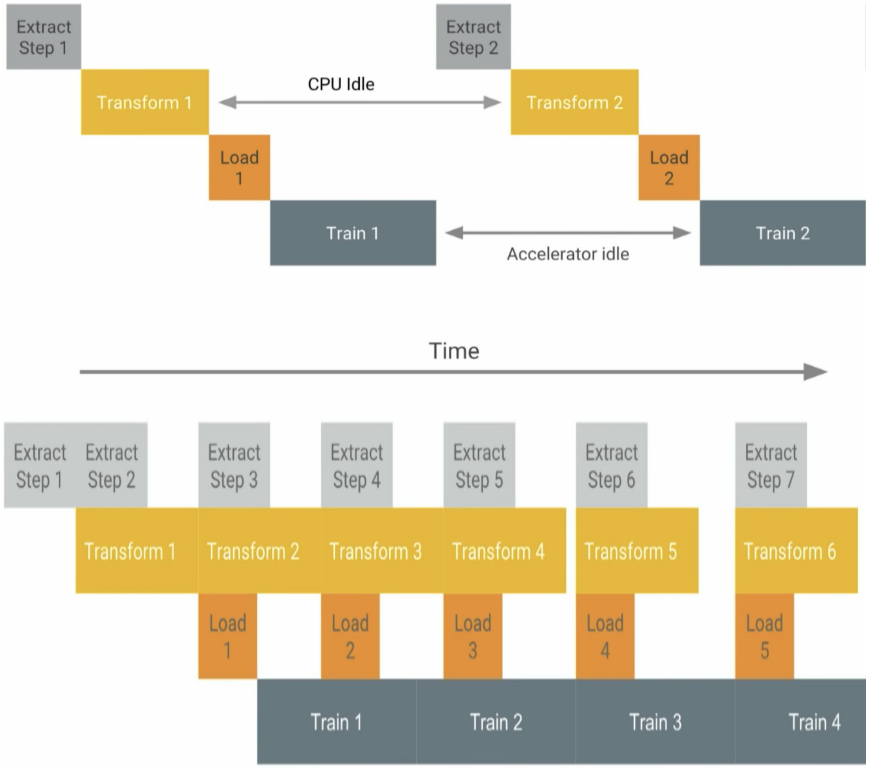
- Scaling deep learning algorithms is critical for Big Data
- Scaling deep learning with Horovod Framework is simple and straightforward
  - Need to add more abstraction in order to improve the performance
- Performance can still be improved through intra-node and internode optimizations

Future Work

Hybrid All_Reduce for Parallel Stochastic Gradient Descent

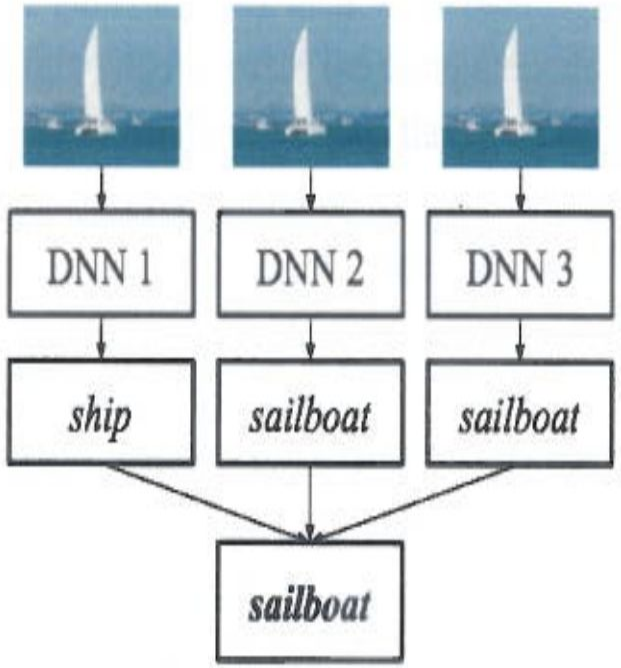


Input / Output Optimization for DNNs

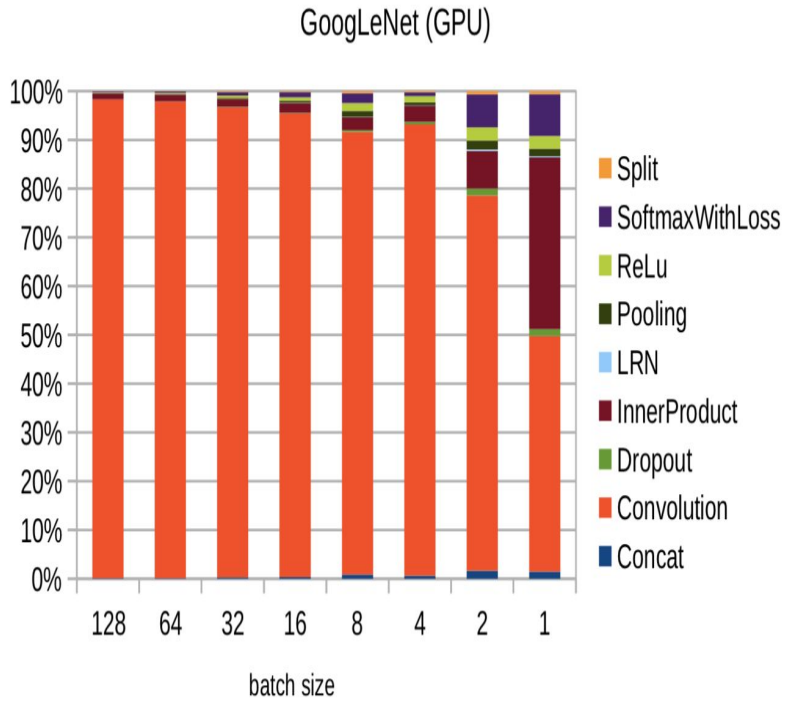


Future Work

Hyper Parameter Optimization (HPO) for DNNs



Data Layout / Memory Optimization for Convolutional DNNs

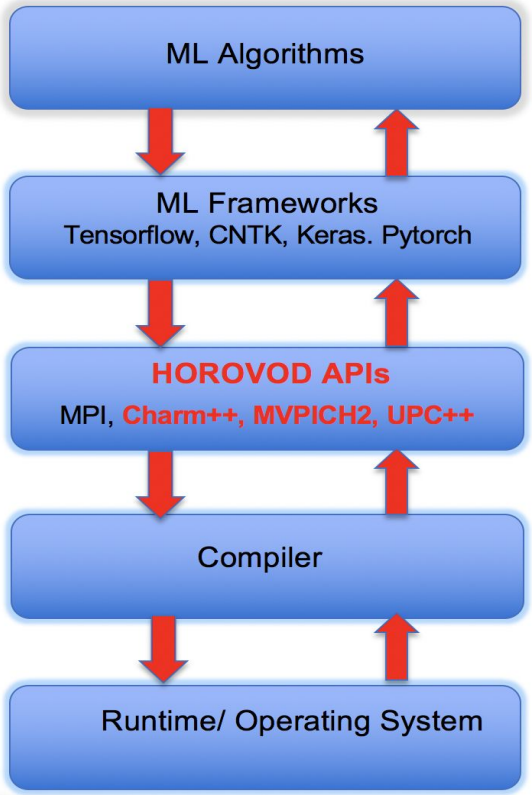


Future Work

Fault Tolerance and Runtime Adaptivity for DNNs

- **Fault tolerance/ Resilience** is a big performance issue in high performance computing
- **Runtime adaptivity** to get the best performance
- **Charm++** is a parallel programming model from UIUC with nice features to handle fault tolerance and runtime adaptivity
- Recently, the group introduced **CharmPy** for distributed data analytics

Unified Software Stack for DNNs



Acknowledgment

We acknowledge the discussion with the Uber AI team (Alexander Sergeev) and Google Tensorflow team

Thank You

- Questions!