Scaling Deep Neural Networks

Abid M. Malik

Computer Science and Mathematics Department

US ATLAS-CSI Workshop, BNL, July 26

BROOKHAVEN NATIONAL LABORATORY

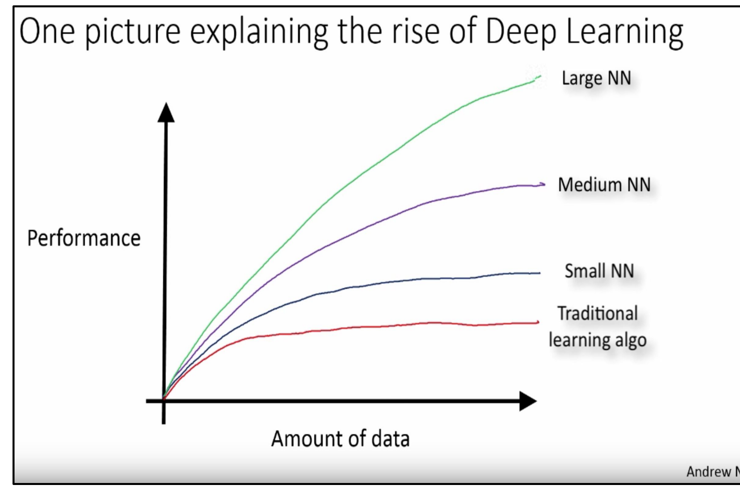
U.S. DEPARTMENT OF ENERGY

Introduction

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

Large simulated datasets are crucial in Particle Physics, just like in Nuclear Physics, Cosmology and other Physical Sciences, for interpreting results of ongoing experiments and for estimating yields of different proposed experimental setups.

Generative Adversarial Networks (GANs) to speed up event simulation in Particle Physics.



TrackML Particle Tracking Challenge

Why we need HPC for Deep Learning?

Jeffery 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! ☺

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

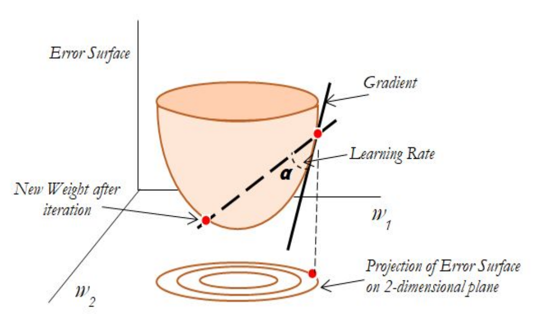
Parallelizing Deep Learning

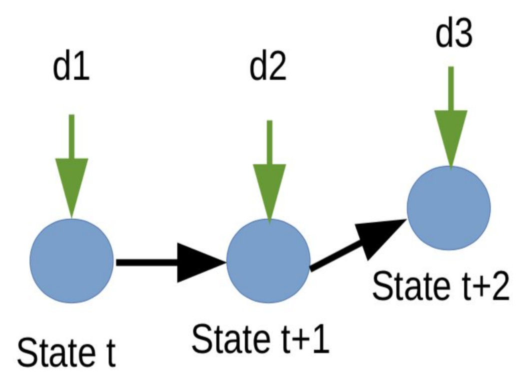
- You parallelize Stochastic Gradient Descent (SGD)
- Parallelizing SGD is very hard. It is inherently sequential algorithm
  1. Start at some state **t** (point in a billion dimensional space)
  2. Introduce **t** to data batch **d1**
  3. Compute an update (based on the objective function)
  4. Apply the update → **t+1**

You improve the performance by:

1. Improving the performance in a node (needs lot of compiler optimizations)– internal parallelism
2. Improving the time between the two states (Input/output and communication is a big challenge)— external parallelism
3. Using large batch size

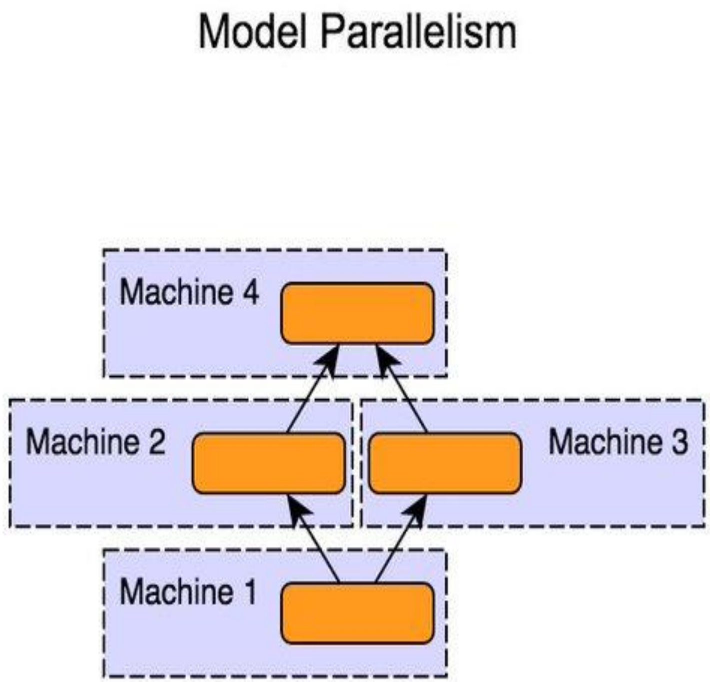
Stochastic Gradient Descent with Batch size "1"

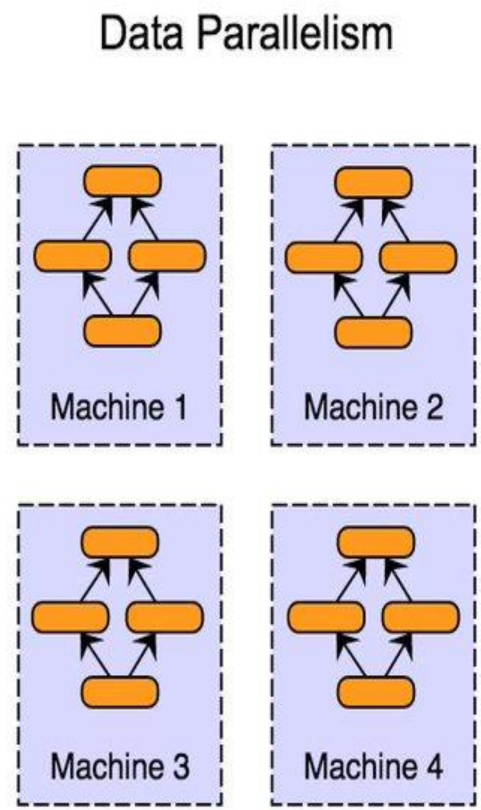




Parallelizing Deep Learning

Types of parallelism

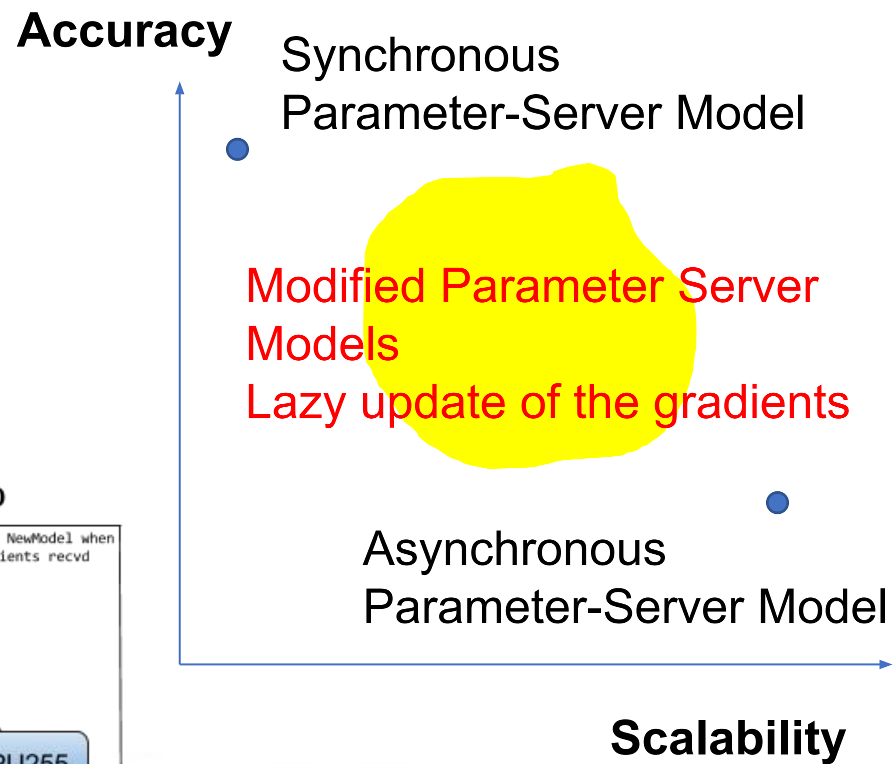


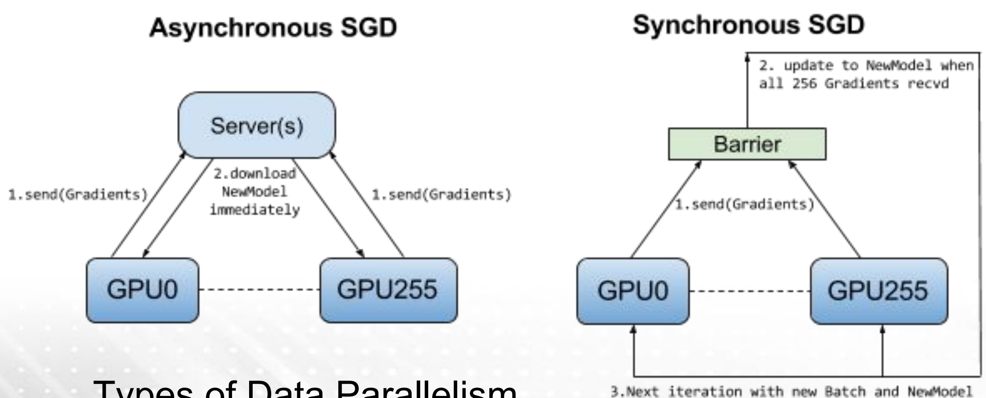


Current parallel deep learning approaches use data parallelism because of better computation and communication ratio

Scalability Performance

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



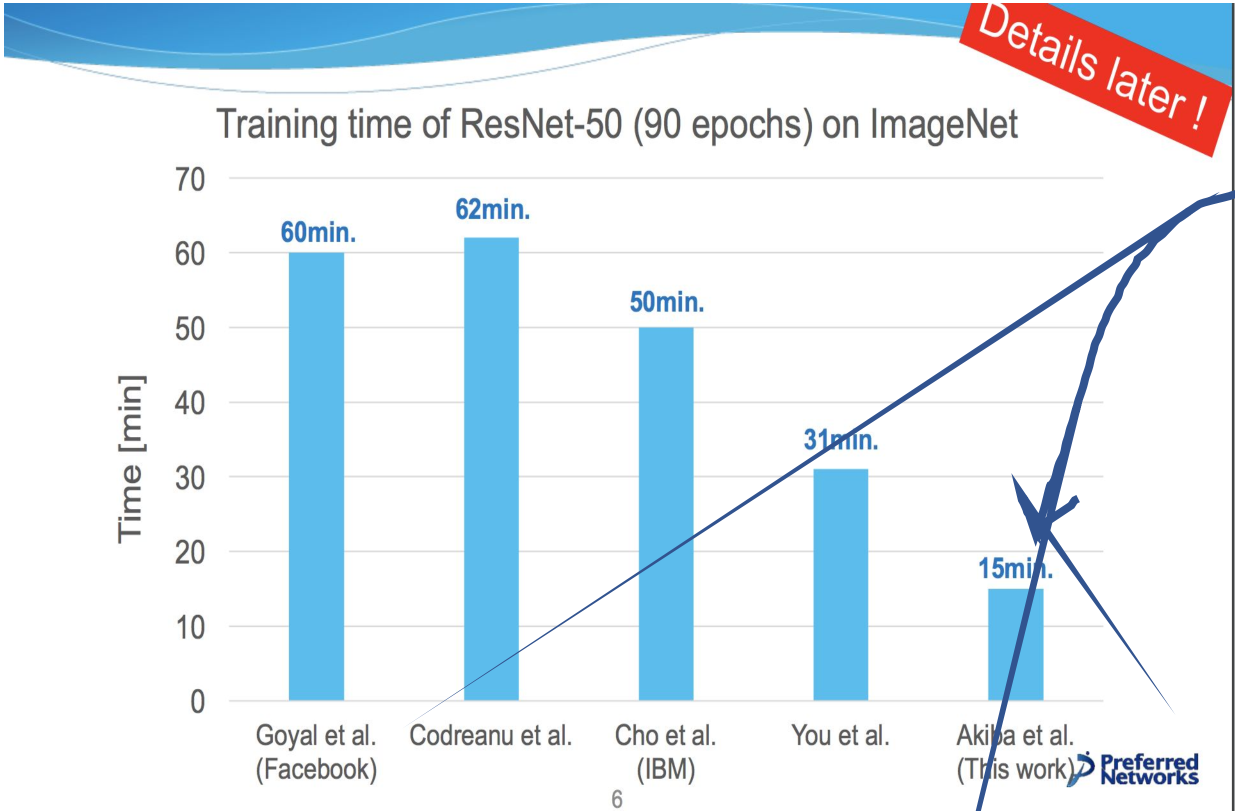


Types of Data Parallelism

U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

State of the Art !



Around 75% accuracy

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

U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

Writing Distributed Deep Learning Algorithms

- Challenges:
  - Deep learning framework or training library must support inter-node and intra-node communication
  - Current 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

Horovod Framework from Uber Inc.

- From Uber AI Engineering Team in 2017
  - Deep learning for self driving cars
- Open-source available at GitHub
- **<u>Provides easy and fast way to do distributed machine learning using Tensorflow, PyTourch, and Keras</u>**
- Efficient synchronized **allreduce** implementation based on Baidu's MPI ring allreduce
- Adopted by **Cray, IBM, Intel**, **Amazon** and **Microsoft** for their distributed machine learning frameworks

HOROVOD

U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

Horovod Framework

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

```
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)
```

1

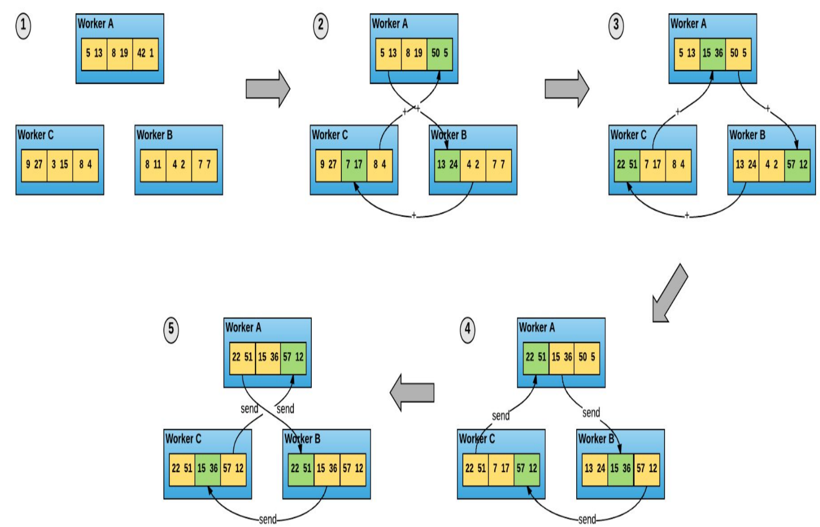
2

3

4

MPI Ring AllReduce

- "Bandwidth Optimal All-reduce Algorithms for Cluster of Workstations" by Patarasuk and Yuan, 2009
- 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
- This algorithm is bandwidth-optimal

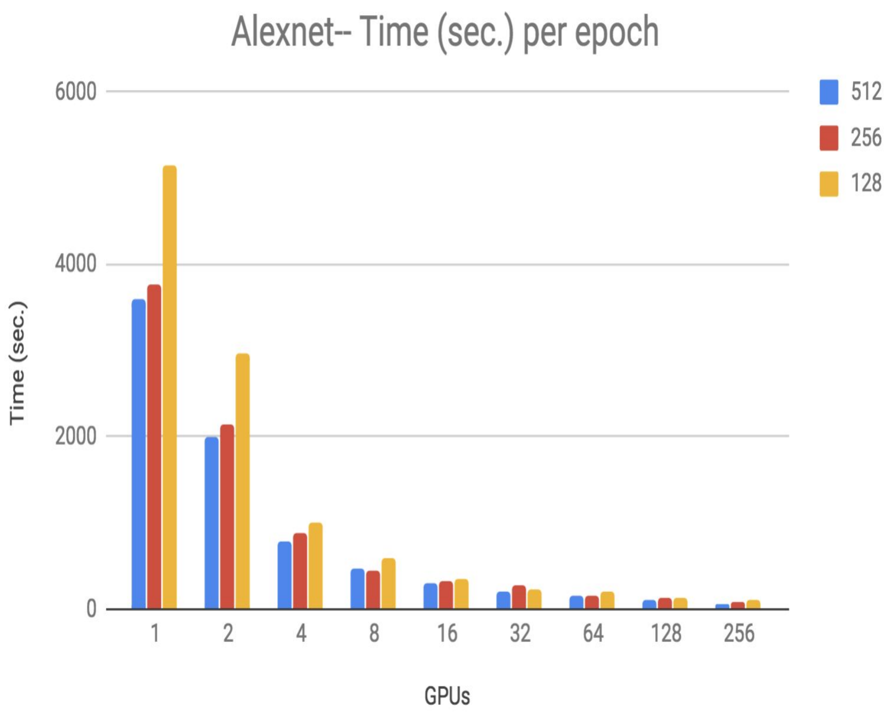


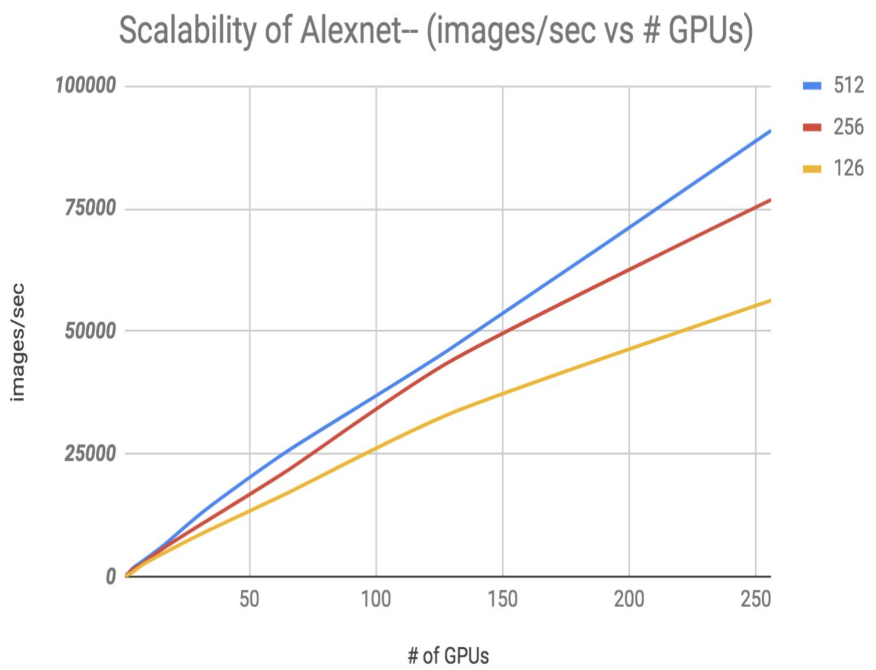
The ring-allreduce algorithm allows worker nodes to average gradients and disperse them to all nodes without the need for a parameter server

Experimentation

- Detailed performance analysis of the Horovod Framework
- We used **AlexNet**, **GoogleNet**, and **ResNet50** implemented in Tensorflow
- We used the ImageNet data set
    - 1.2 million images
- We used Nvidia K80 and P100 GPUs on the Institutional Cluster at BNL
- 124 worker nodes
- InfiniBand EDR connectivity

AlexNet using Horovod

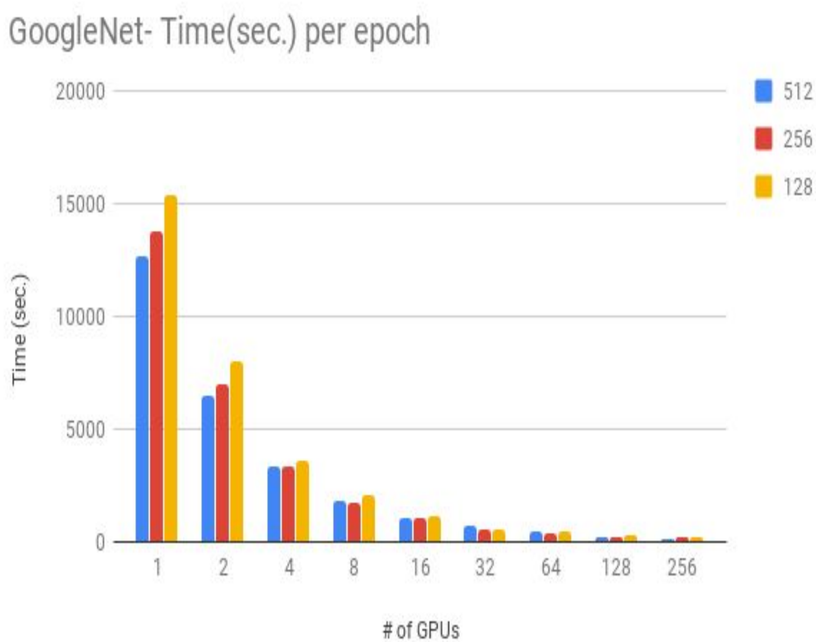


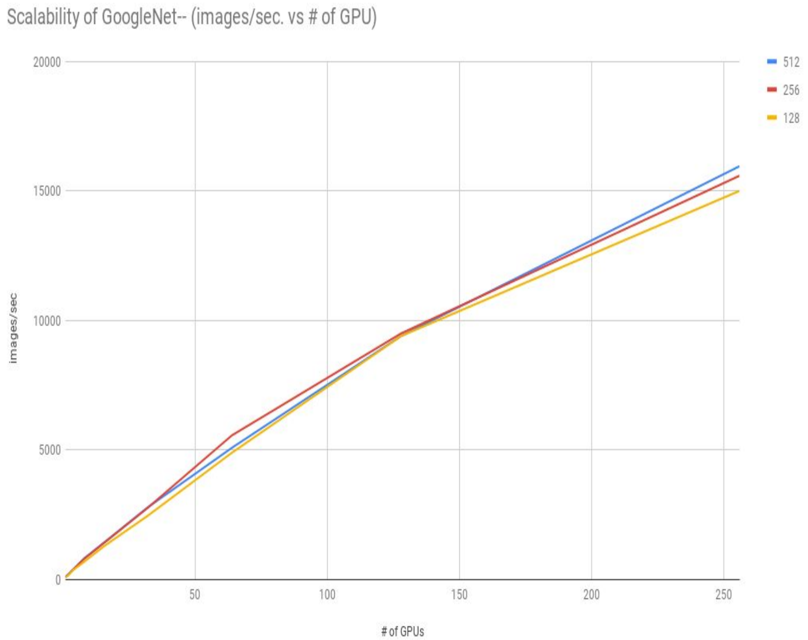


U.S. DEPARTMENT OF ENERGY

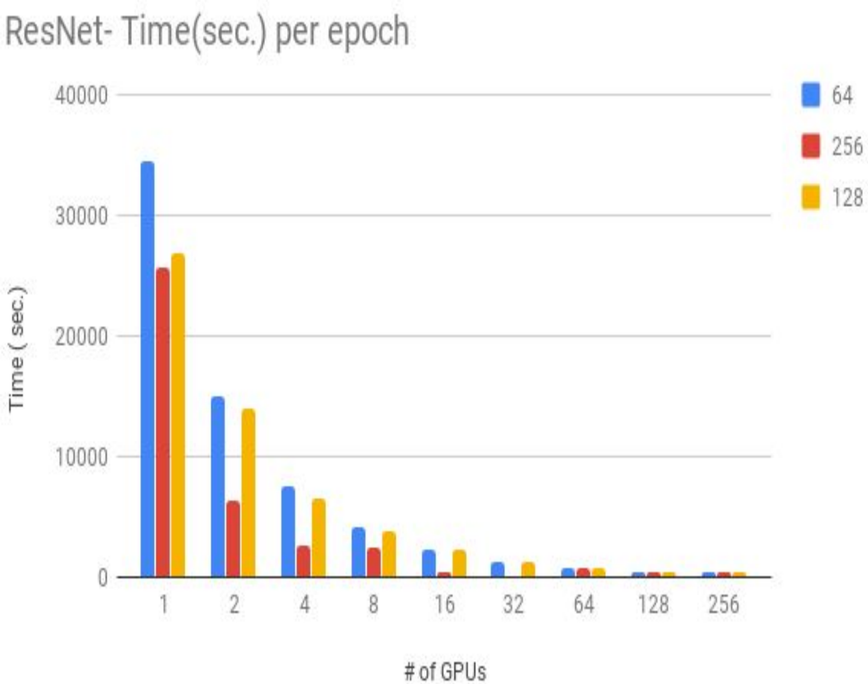
BROOKHAVEN NATIONAL LABORATORY

GoogleNet using Horovod





ResNet50 using Horovod

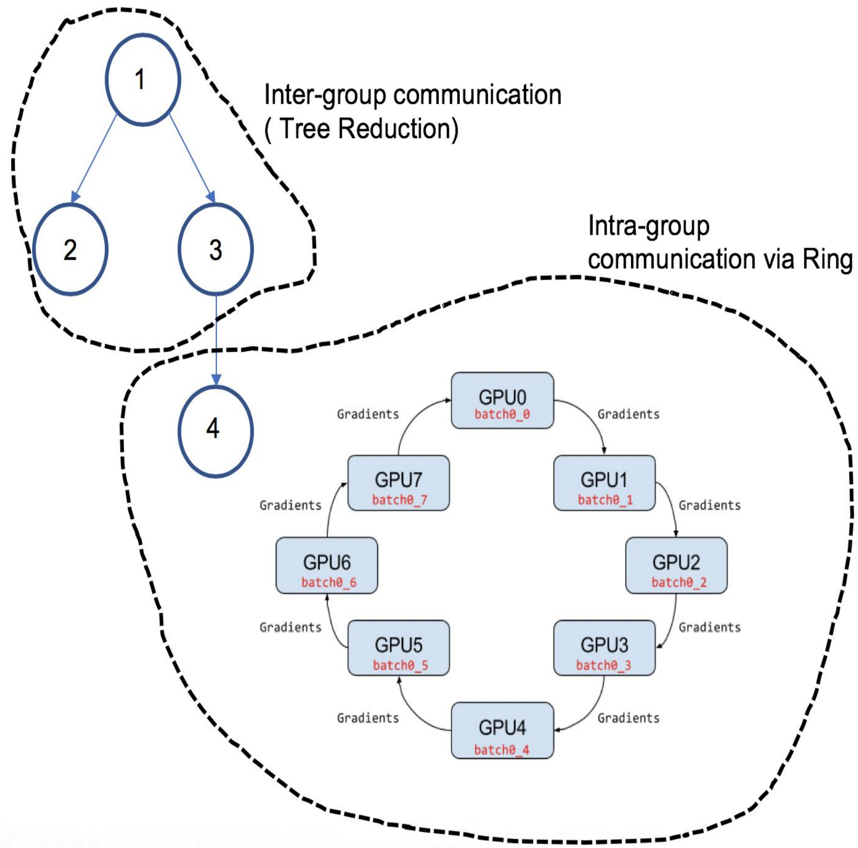


U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

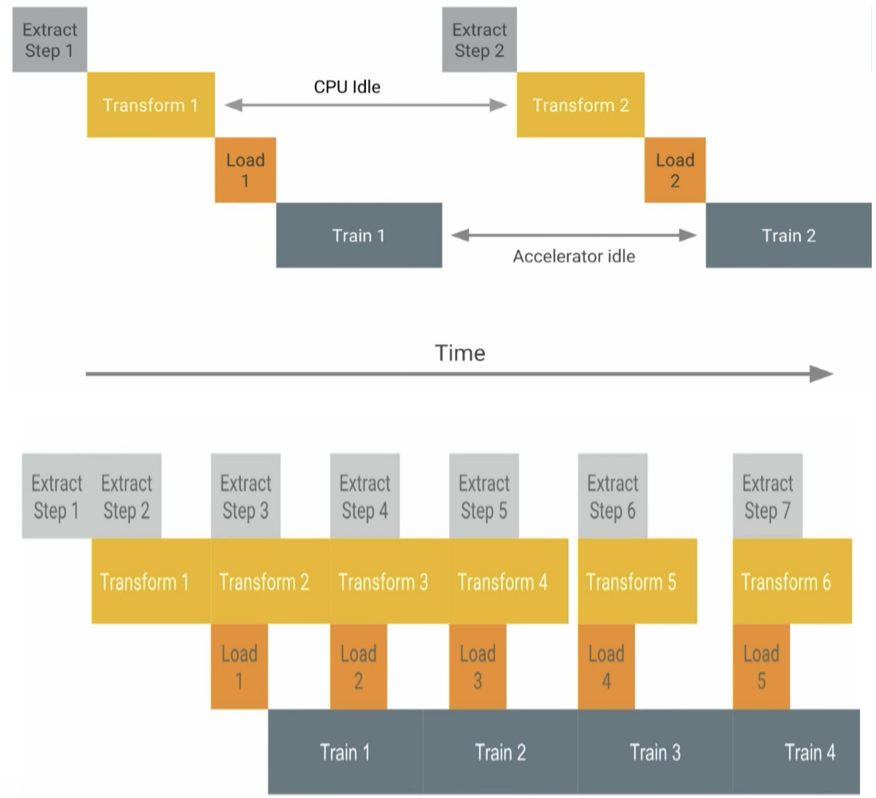
Hybrid MPI All_Rrduce for Parallel Stochastic Gradient Descent

- The ring algorithm doesn’t scale linearly after 512 GPUs
- Hybrid approach that can scale well on a machine like “Summit”



Input / Output Optimization for DNNs

- Input / Output Pipeline
  - **Extract** data from storage
  - **Transform** data to prepare for training
  - **Load** data into the accelerator
  - **Train** the model using the data

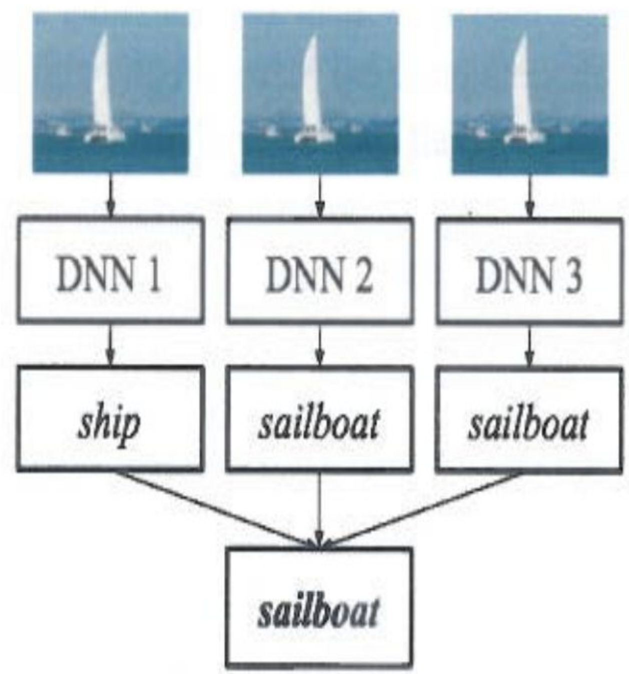


U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

Hyperparameter Optimization for DNNs

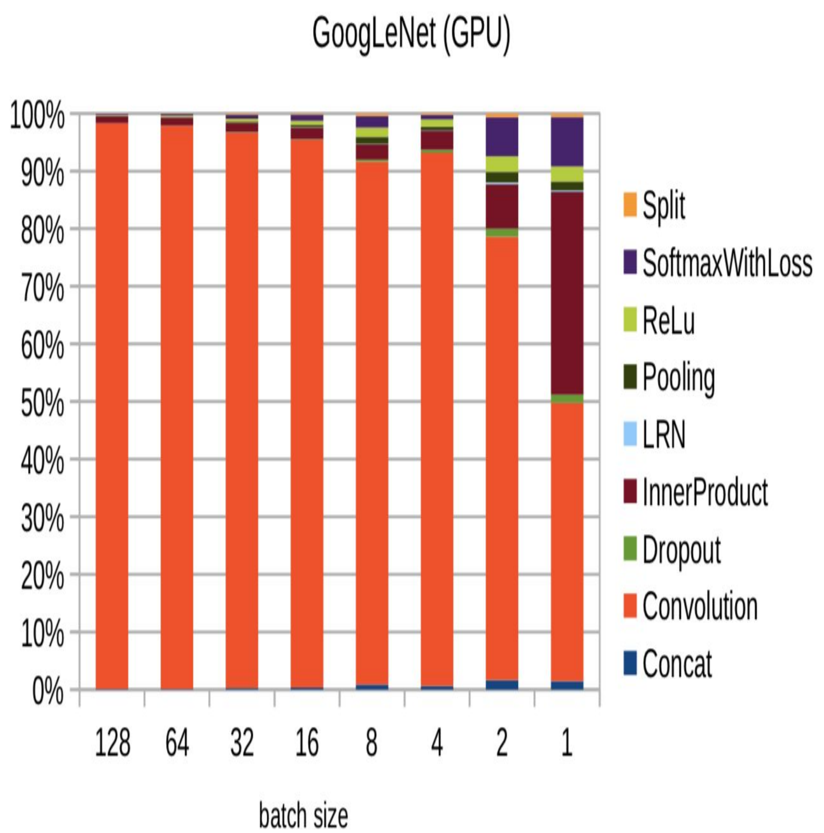
- For a given throughput, you need to find parameter for better convergence
- Using an ensemble workflow framework for training different models and then selecting the best



A DNN ensemble of size 3 for an image of a sailboat

Data Layout / Memory Optimization for Convolutional DNNs

- Internal optimization
- Matrix matrix multiplications
- cuDNN and other cuBLAS

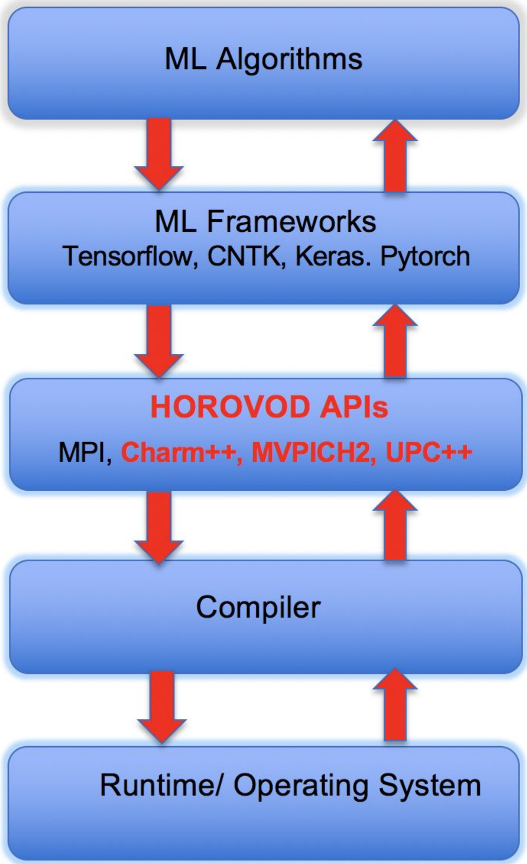


Fault Tolerance and Runtime Adaptivity for DNNs

- Fault tolerance 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

- We need to treat Deep Neural Nets as first class citizen
- Special language and optimizations features that can help write a domain scientist to write ML code
- Horovod glues together the ML frameworks and different distributed implementations



Conclusion

- 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
- Unified Software Stack would help end user write distributed deep learning algorithms

Thank You

- Questions!