MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

Follow us on

https://twitter.com/mvapich

HiDL
High-Performance Deep Learning

Distributed Deep Neural Network Training using MVAPICH2

Talk at OSU Booth SC '22

Arpan Jain

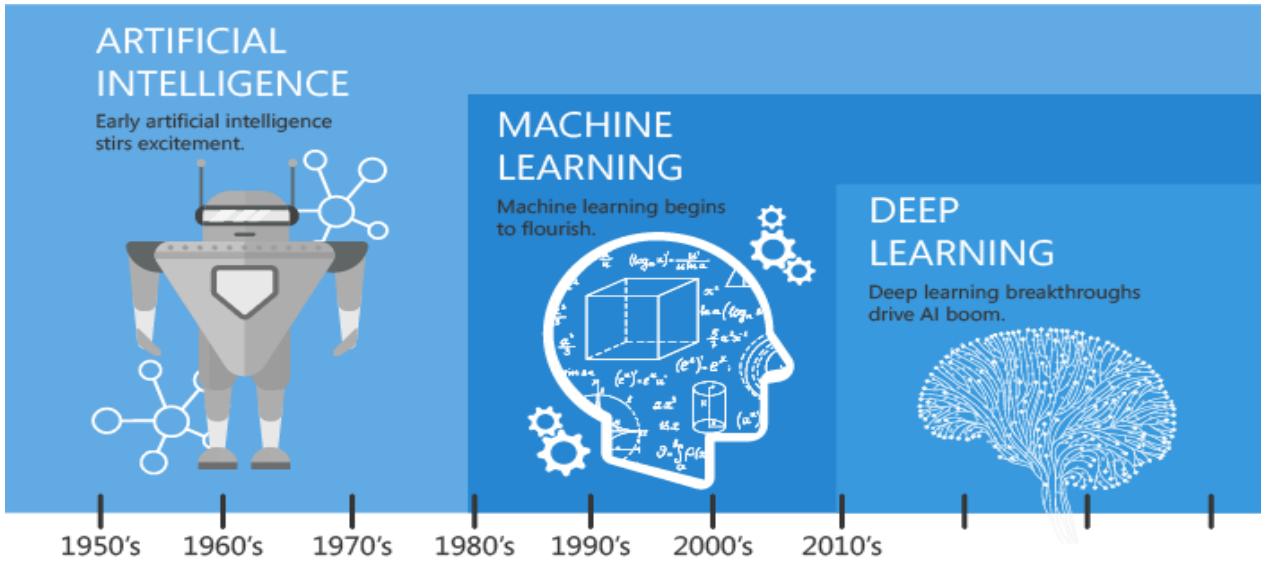
The Ohio State University

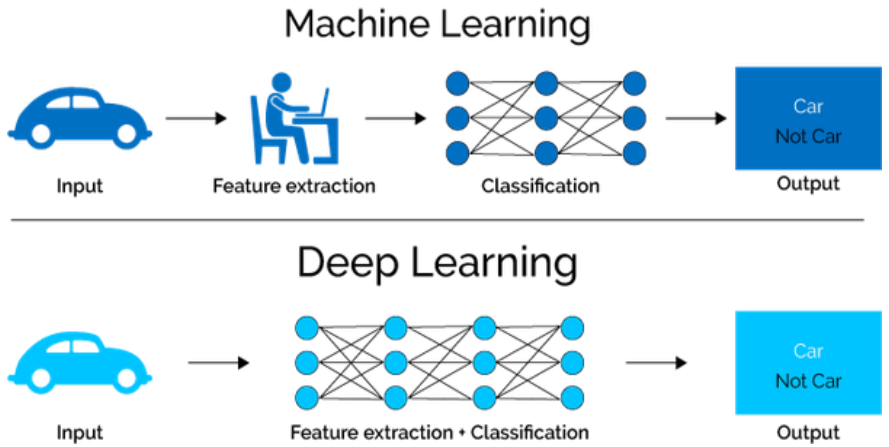
jain.575@osu.edu

http://u.osu.edu/jain.575

What is Machine Learning and Deep Learning?

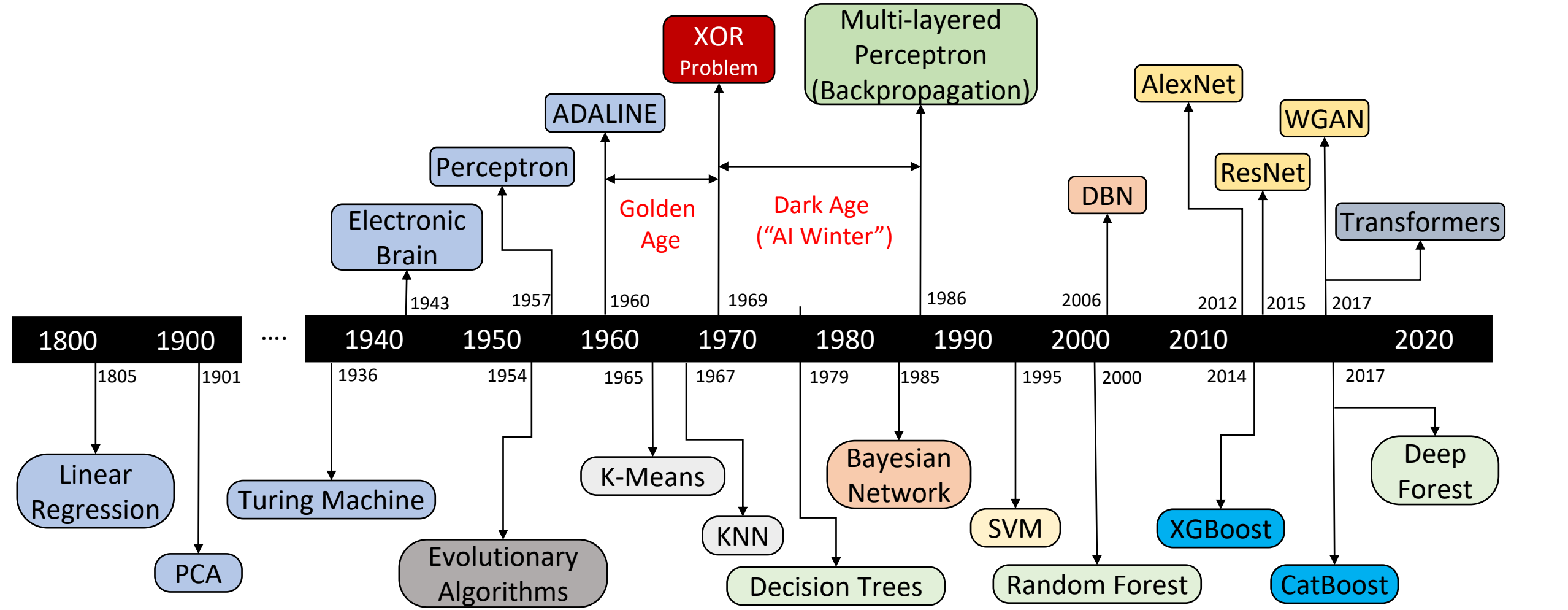
- Machine Learning (ML)
  - "the study of computer algorithms to improve automatically through experience and use of data"
- Deep Learning (DL) – a subset of ML
  - Uses Deep Neural Networks (DNNs)
  - **Perhaps, the most revolutionary subset!**
- Based on learning data representation
- DNN Examples: Convolutional Neural Networks, Recurrent Neural Networks, Hybrid Networks
- Data Scientist or Developer Perspective for using DNNs
  1. Identify DL as solution to a problem
  2. Determine Data Set
  3. Select Deep Learning Algorithm to Use
  4. Use a large data set to train an algorithm





Courtesy: https://hackernoon.com/difference-between-artificial-intelligence-machine-learning-and-deep-learning-1pcv3zeg, https://blog.dataiku.com/ai-vs.-machine-learning-vs.-deep-learning, https://en.wikipedia.org/wiki/Machine_learning

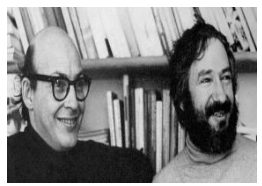
History: Milestones in the Development of ML/DL



A. Legendre – J. Gauss K. Pearson

A. Turing S. McCulloch – W. Pitts F. Rosenblatt

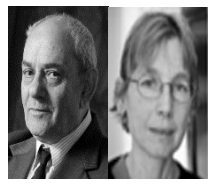
B. Widrow – M. Hoff



M. Minsky – S. Papert

J. Pearl

D. Rumelhart – G. Hinton – R. Wiliams



V. Vapnik– C. Cortes

A. Krizhevsky

A. Ng

Y. LeCun

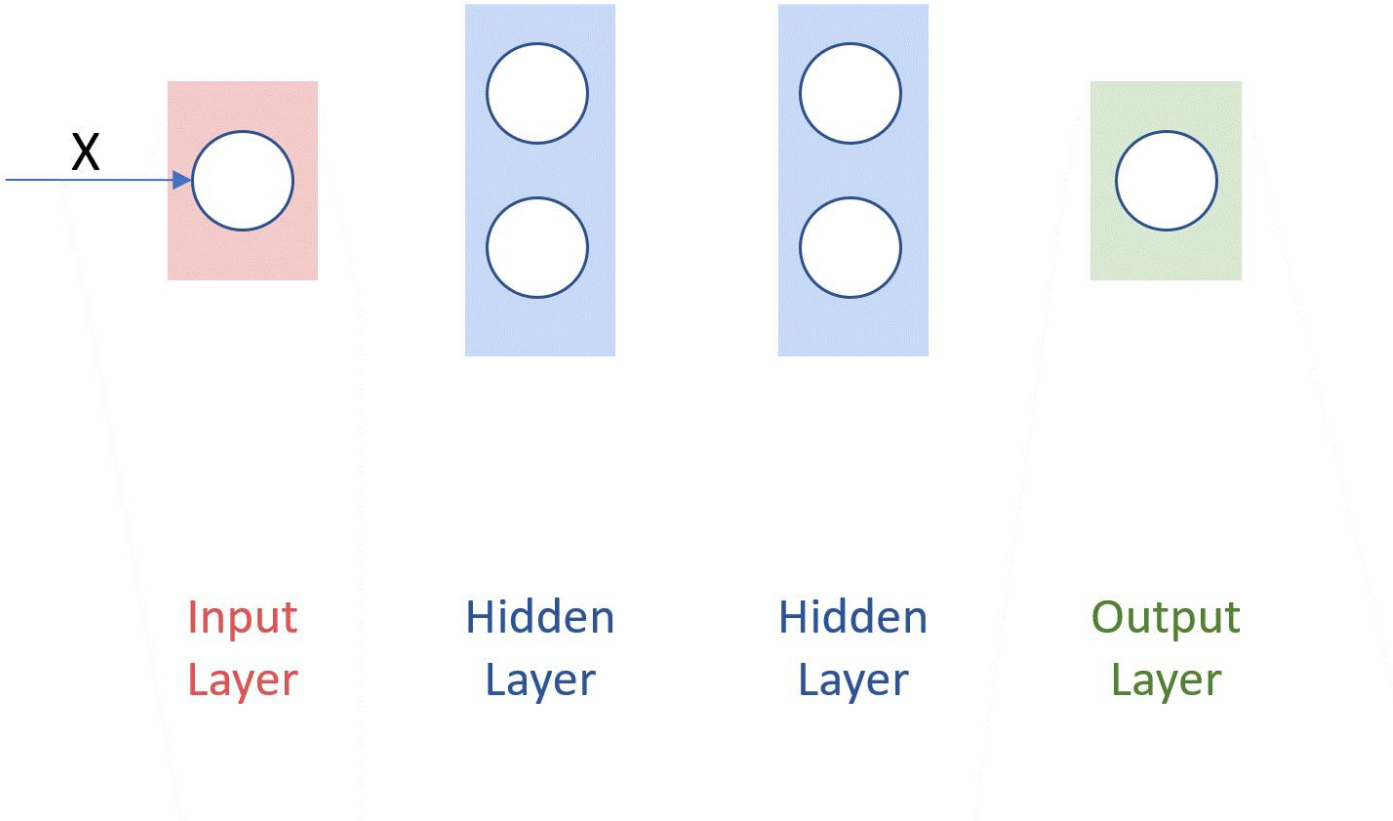
Y. Bengio

Outline

- Introduction
- Deep Learning
  - Deep Neural Networks
  - Distributed Deep Learning
  - DL Solutions
- Conclusion

Understanding the Deep Neural Network Concepts

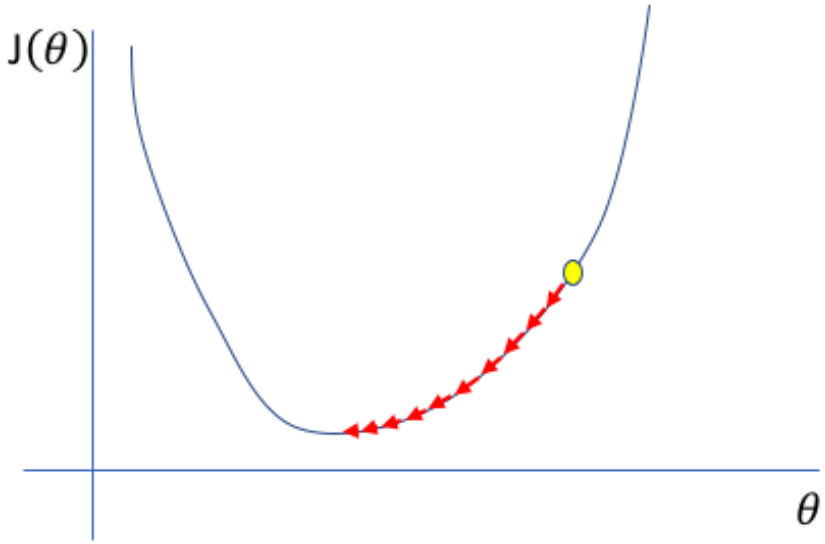
- Example of a 3-layer Deep Neural Network (DNN) – (input layer is not counted)



Courtesy: http://cs231n.github.io/neural-networks-1/

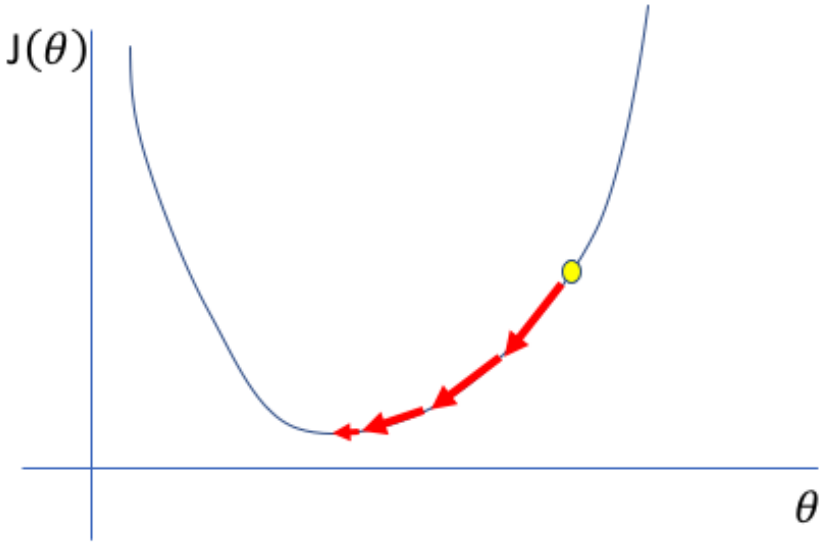
Essential Concepts: Learning Rate (α)

Too low



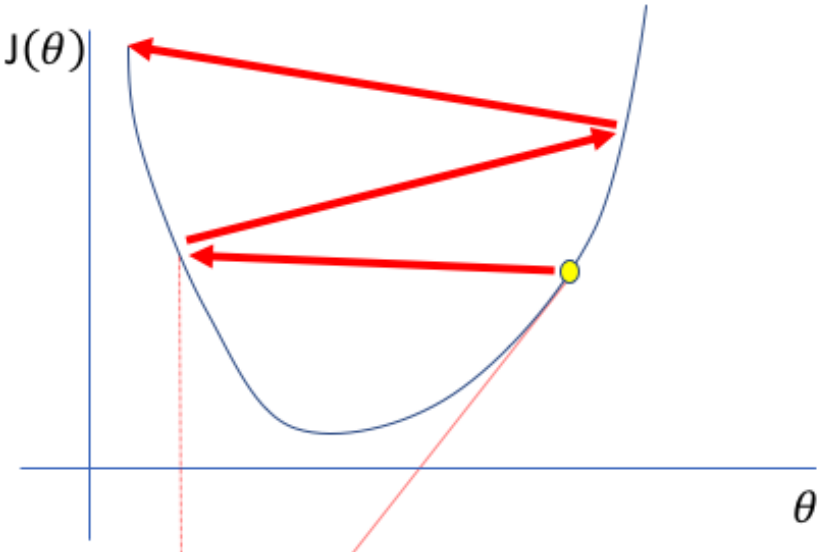
A small learning rate requires many updates before reaching the minimum point

Just right



The optimal learning rate swiftly reaches the minimum point

Too high

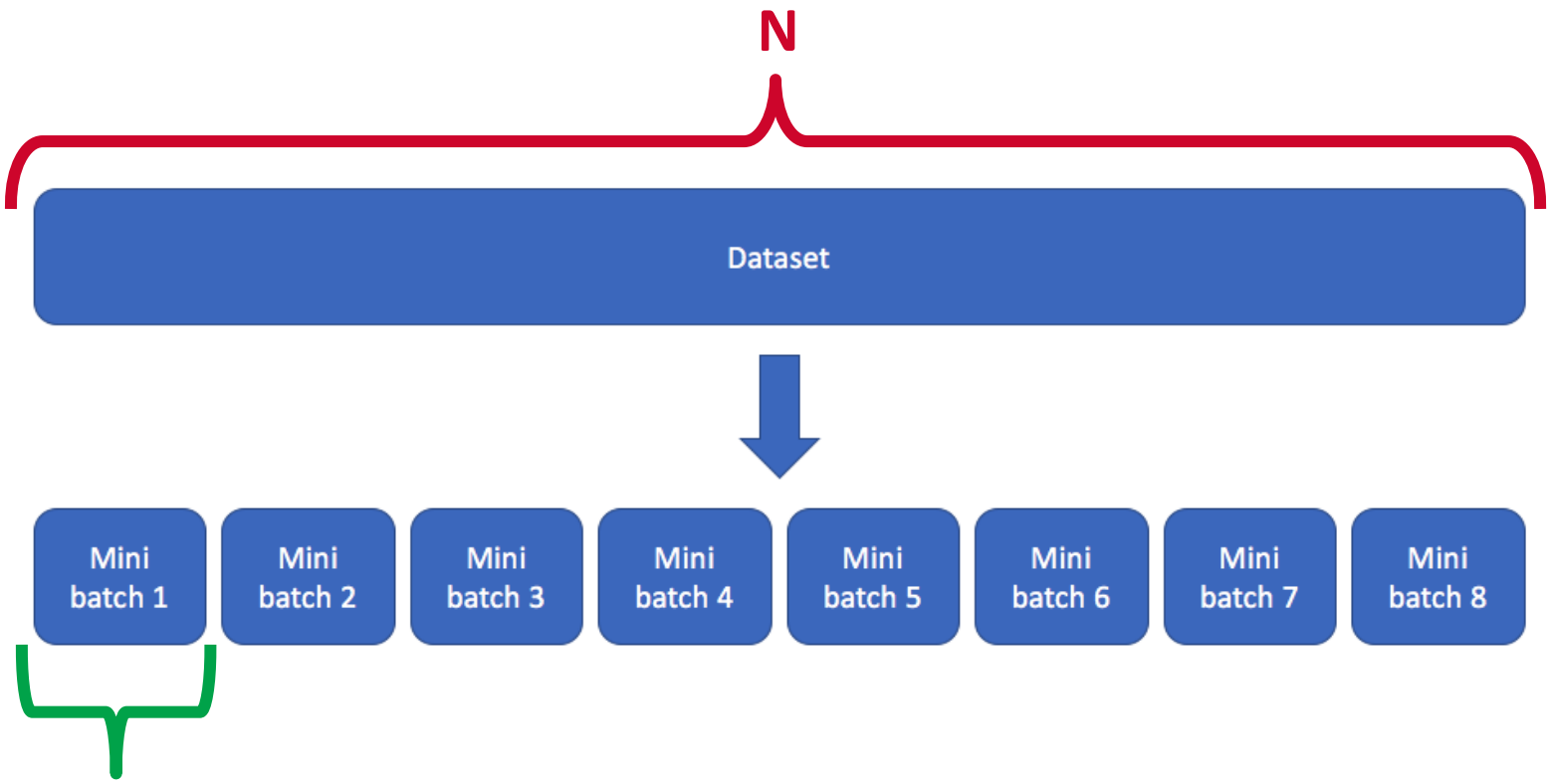


Too large of a learning rate causes drastic updates which lead to divergent behaviors

Courtesy: https://www.jeremyjordan.me/nn-learning-rate/

Essential Concepts: Batch Size

- Batched Gradient Descent
  - Batch Size = **N**
- Stochastic Gradient Descent
  - Batch Size = **1**
- Mini-batch Gradient Descent
  - Somewhere in the middle
  - Common:
    - **Batch Size** = 64, 128, 256, etc.
- Finding the optimal batch size will yield the fastest learning.



*One full pass over **N** is called an **epoch** of training*

Courtesy: https://www.jeremyjordan.me/gradient-descent/

Outline

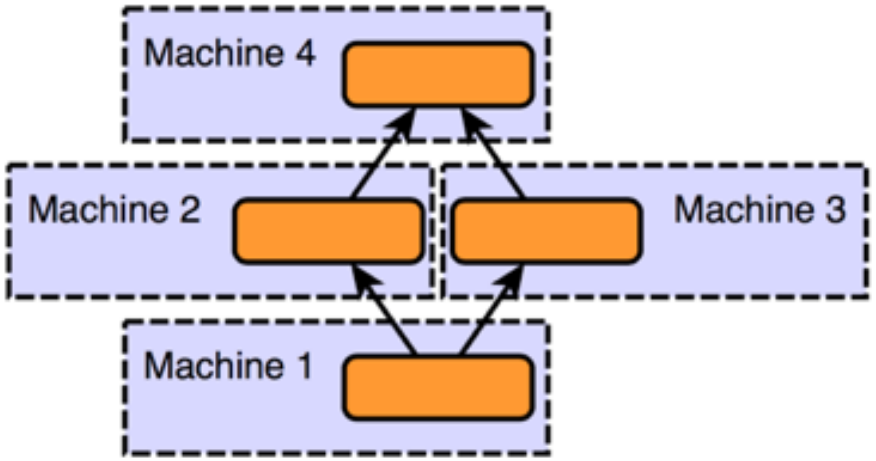
- Introduction
- Deep Learning
  - Deep Neural Networks
  - Distributed Deep Learning
  - DL Solutions
- Conclusion

The Need for Parallel and Distributed Training

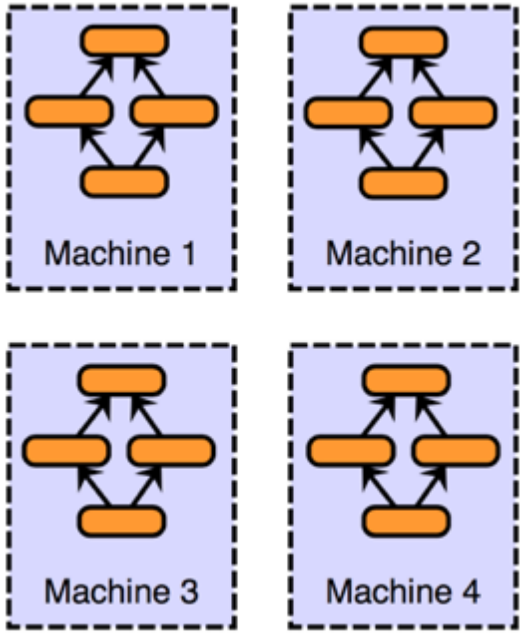
- Why do we need Parallel Training?
- Larger and Deeper models are being proposed
  - **AlexNet** -> **ResNet** -> **NASNet – AmoebaNet**
  - DNNs require a lot of memory and a lot of computation
  - Larger models cannot fit a GPU's memory
- Single GPU training cannot keep up with ever-larger models
- Community has moved to multi-GPU training
- Multi-GPU in one node is good but there is a limit to Scale-up (8-16 GPUs)
- **Multi-node (Distributed or Parallel) Training is necessary!!**

Parallelization Strategies

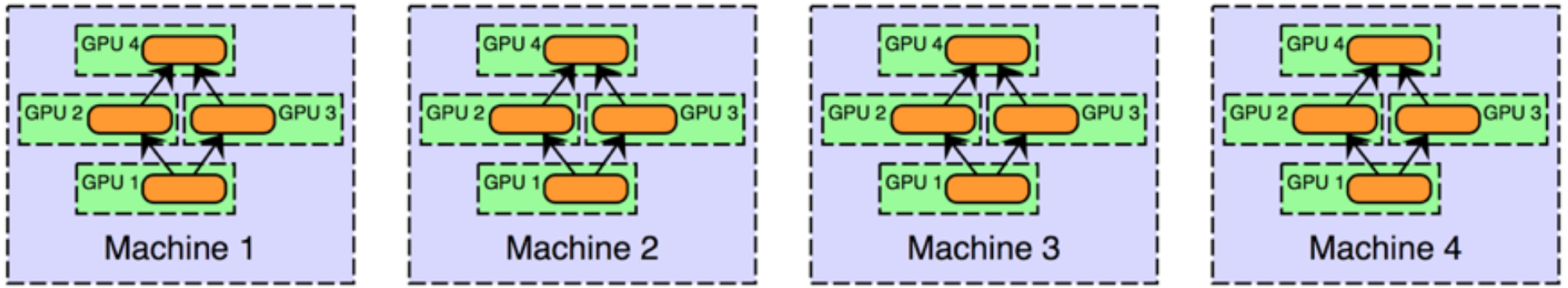
- Some parallelization strategies..
  - Data Parallelism or Model Parallelism
  - Hybrid Parallelism



Model Parallelism



Data Parallelism

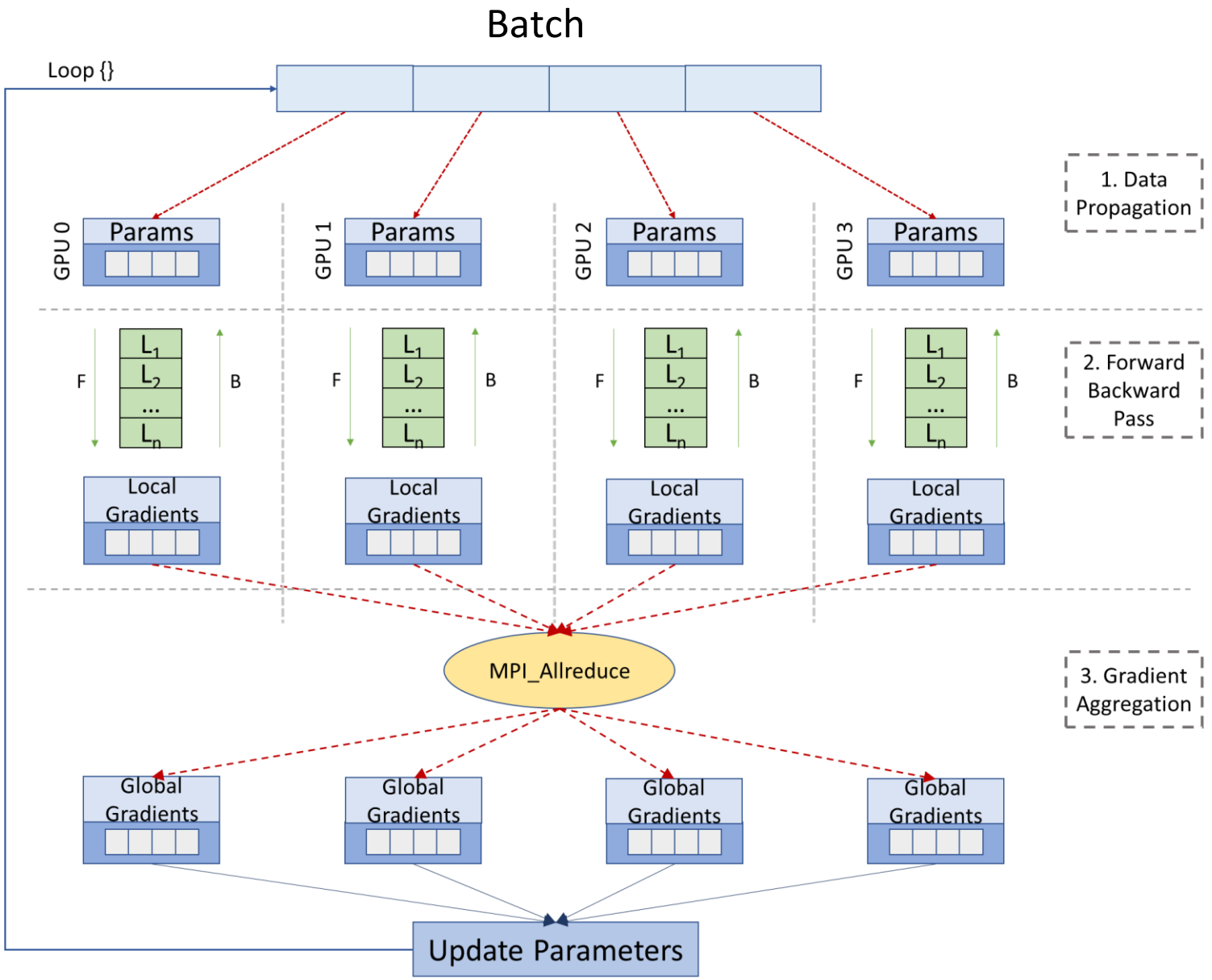


Hybrid (Model and Data) Parallelism

Courtesy: http://engineering.skymind.io/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks

Data Parallelism and MPI Collectives

- Step1: Data Propagation
  - Distribute the Data among GPUs
- Step2: Forward Backward Pass
  - Perform forward pass and calculate the prediction
  - Calculate Error by comparing prediction with actual output
  - Perform backward pass and calculate gradients
- Step3: Gradient Aggregation
  - Call MPI_Allreduce to reduce the local gradients
  - Update parameters locally using global gradients

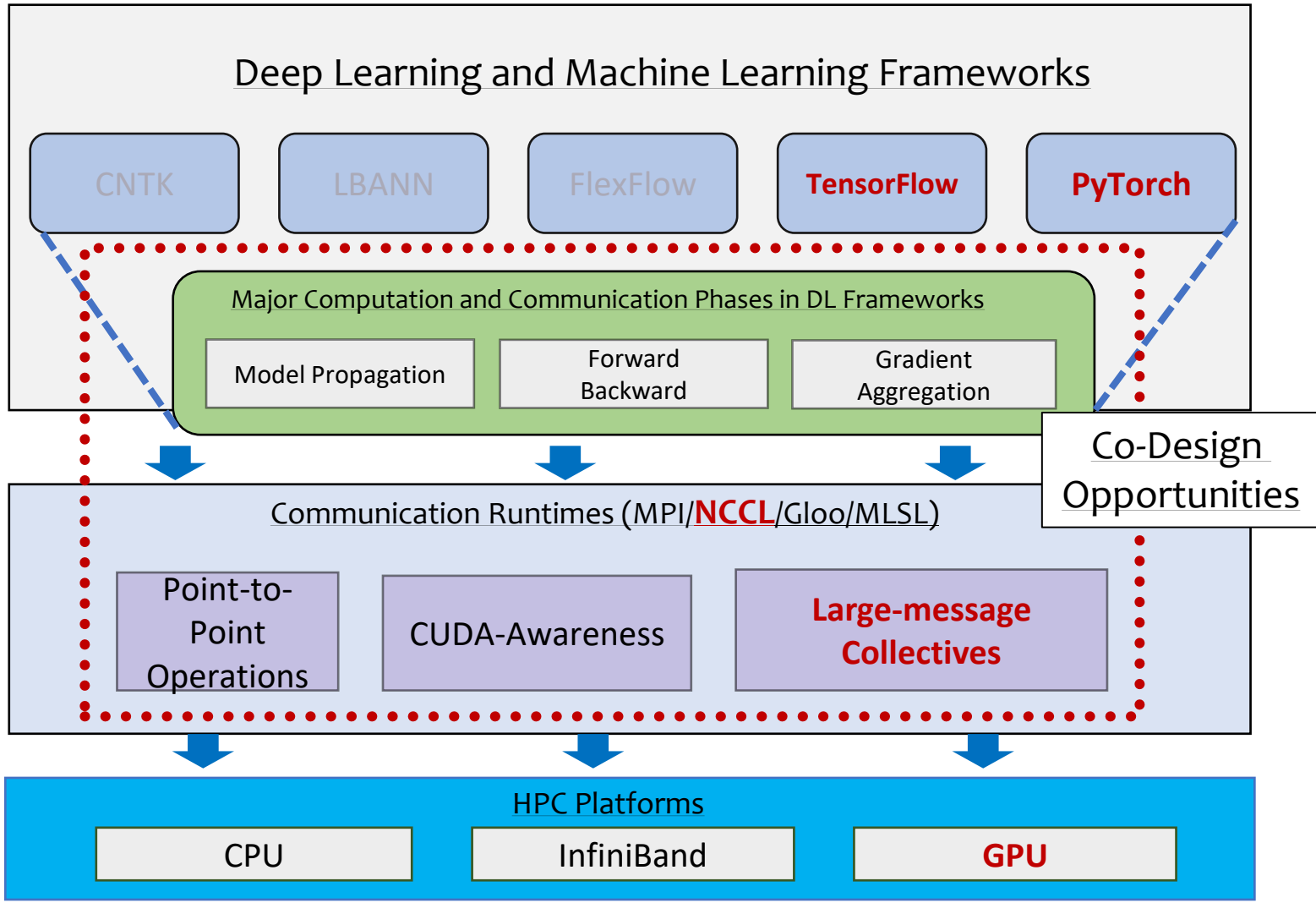


Outline

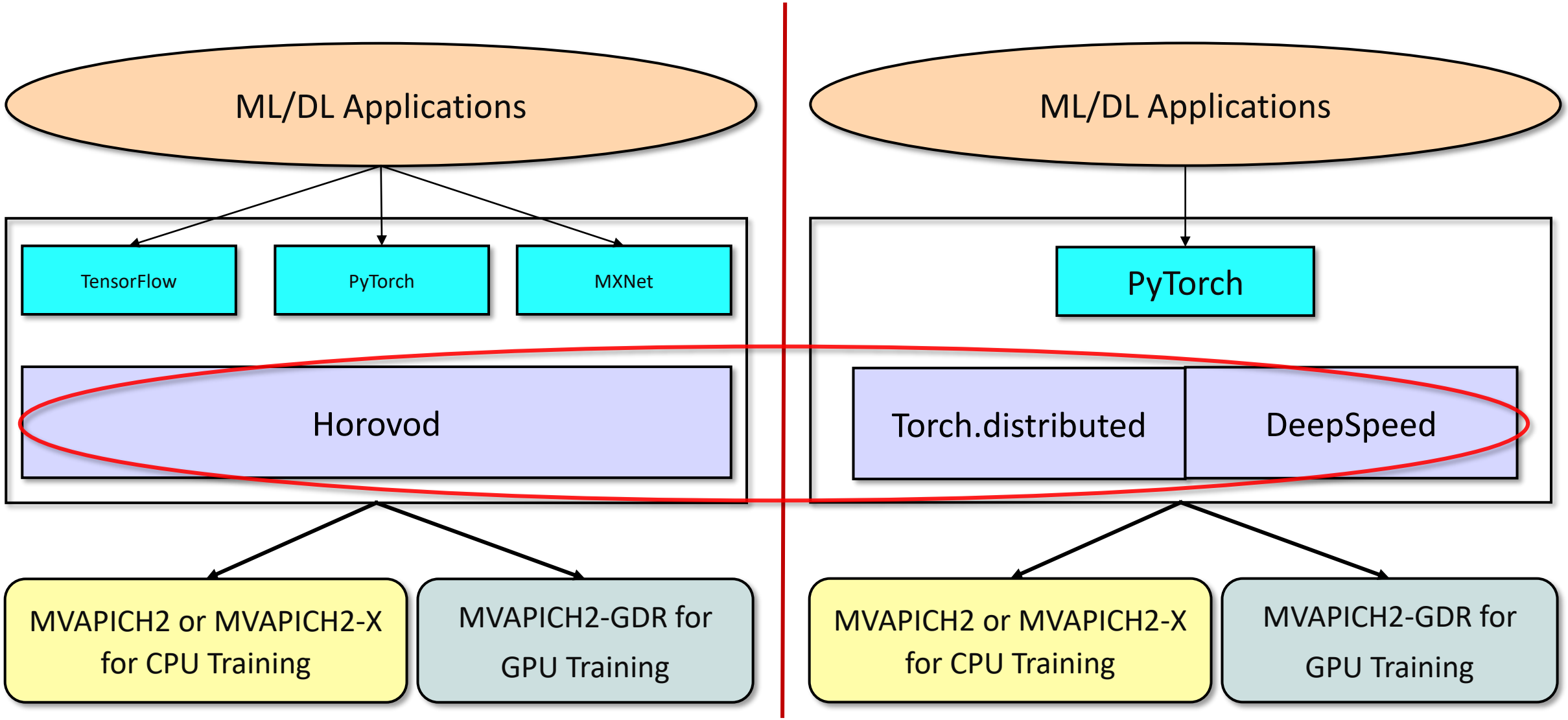
- Introduction
- Deep Learning
  - Deep Neural Networks
  - Distributed Deep Learning
  - DL Solutions
- Conclusion

Solutions and Case Studies: Exploiting HPC for DL

- **Data Parallelism**
- Model-Parallelism



MVAPICH2 (MPI)-driven Infrastructure for ML/DL Training



More details available from: http://hidl.cse.ohio-state.edu

Install Horovod with MVAPICH2-X and MVAPICH2-GDR

Command to install Horovod with MVAPICH2-X

```
$ HOROVOD_WITH_MPI=1 pip install --no-cache-dir horovod
```

Command to install Horovod with MVAPICH2-GDR

```
$ HOROVOD_GPU_ALLREDUCE=MPI HOROVOD_CUDA_HOME=/opt/cuda/11.3 HOROVOD_WITH_MPI=1 pip
install --no-cache-dir horovod
```

Run PyTorch on a single GPU

+ python pytorch_synthetic_benchmark.py --batch-size 64 --num-iters=5

```
.
----------------------------------------------------------------
.
Model: resnet50
Batch size: 64
Number of GPUs: 1
Running warmup...
Running benchmark...
Iter #0: 333.9 img/sec per GPU
Iter #1: 334.2 img/sec per GPU
Iter #2: 333.9 img/sec per GPU
Iter #3: 333.8 img/sec per GPU
Iter #4: 333.9 img/sec per GPU
Img/sec per GPU: 334.0 +-0.2
----------------------------------------
Total img/sec on 1 GPU(s): 334.0 +-0.2
----------------------------------------
```

V100

Run PyTorch on two nodes with 1 GPU/node (using MVAPICH2-GDR)

+ mpirun_rsh -np 2 gpu11 gpu12 MV2_USE_CUDA=1 MV2_CPU_BINDING_POLICY=hybrid MV2_HYBRID_BINDING_POLICY=spread MV2_USE_RDMA_CM=0 MV2_GPUDIRECT_GDRCOPY_LIB=/opt/gdrcopy2.0/lib64/libgdrapi.so LD_PRELOAD=mv2-gdr/lib/libmpi.so python pytorch_synthetic_benchmark.py --batch-size 64 --num-iters=5

.

.

V100

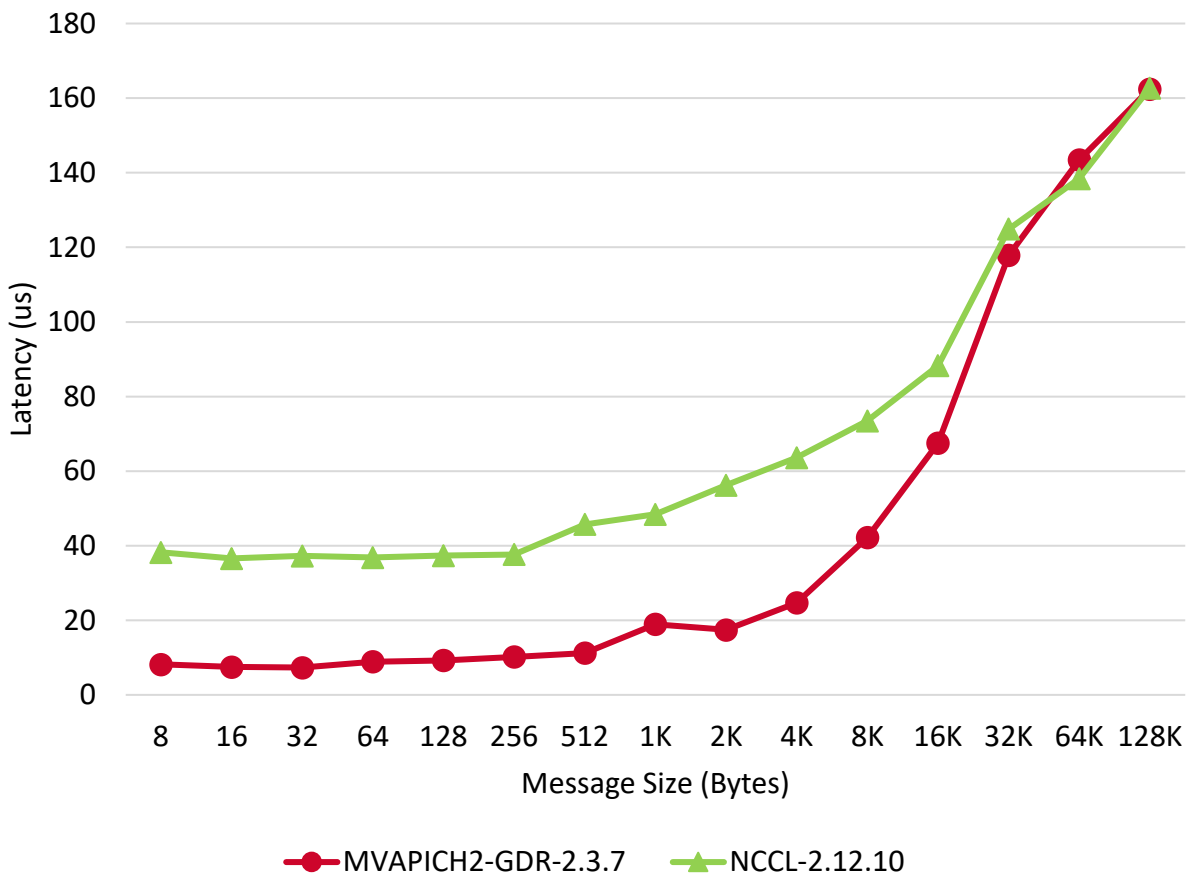
```
Model: resnet50
Batch size: 64
Number of GPUs: 2
Running warmup...
Running benchmark...
Iter #0: 317.0 img/sec per GPU
Iter #1: 314.9 img/sec per GPU
Iter #2: 315.4 img/sec per GPU
Iter #3: 318.0 img/sec per GPU
Iter #4: 316.7 img/sec per GPU
Img/sec per GPU: 316.4 +-2.2
-----------------------------------------
Total img/sec on 2 GPU(s): 632.8 +-4.3
-----------------------------------------
```

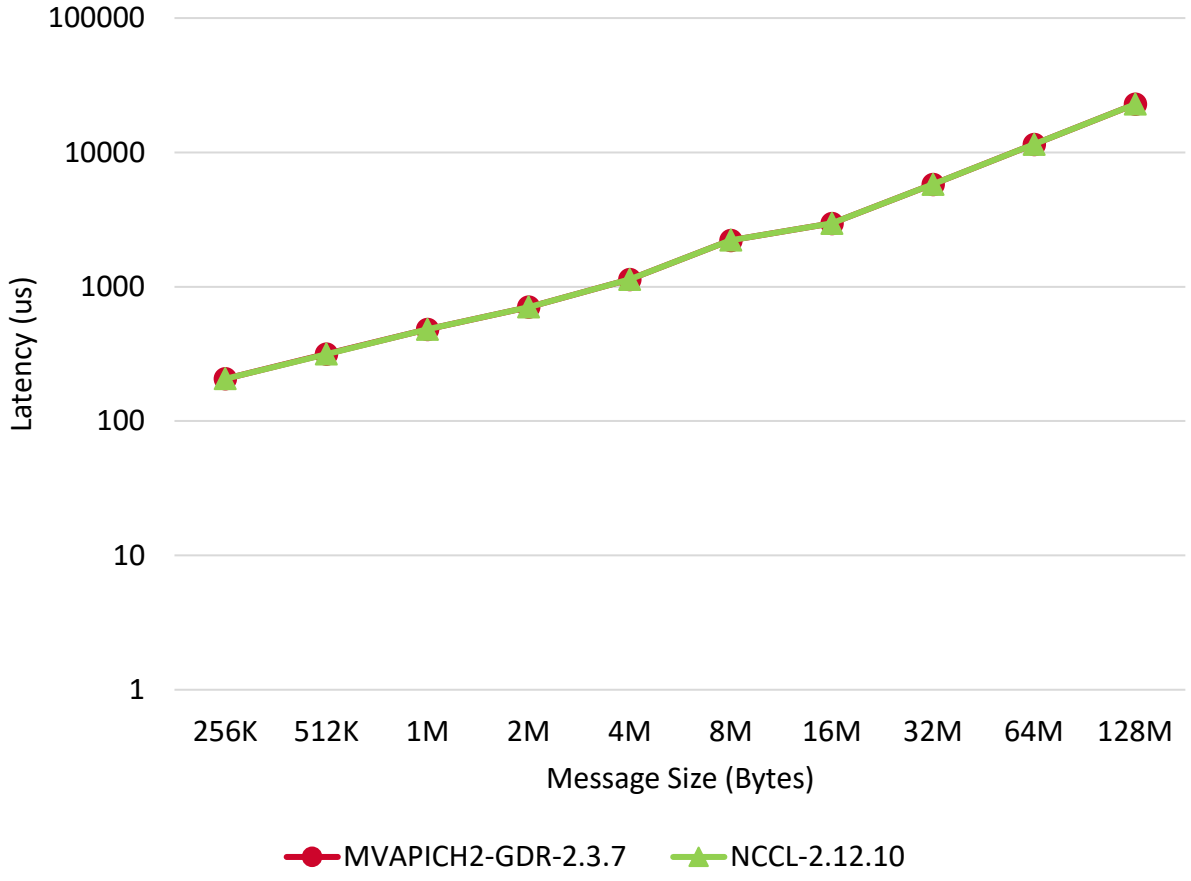
~1.89X on 2 GPUs

MVAPICH2-GDR vs. NCCL2 – Allreduce Operation (OSC Pitzer)

- **Optimized designs in MVAPICH2-GDR offer better/comparable performance for most cases**
- **MPI_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) on OSC Pitzer system**

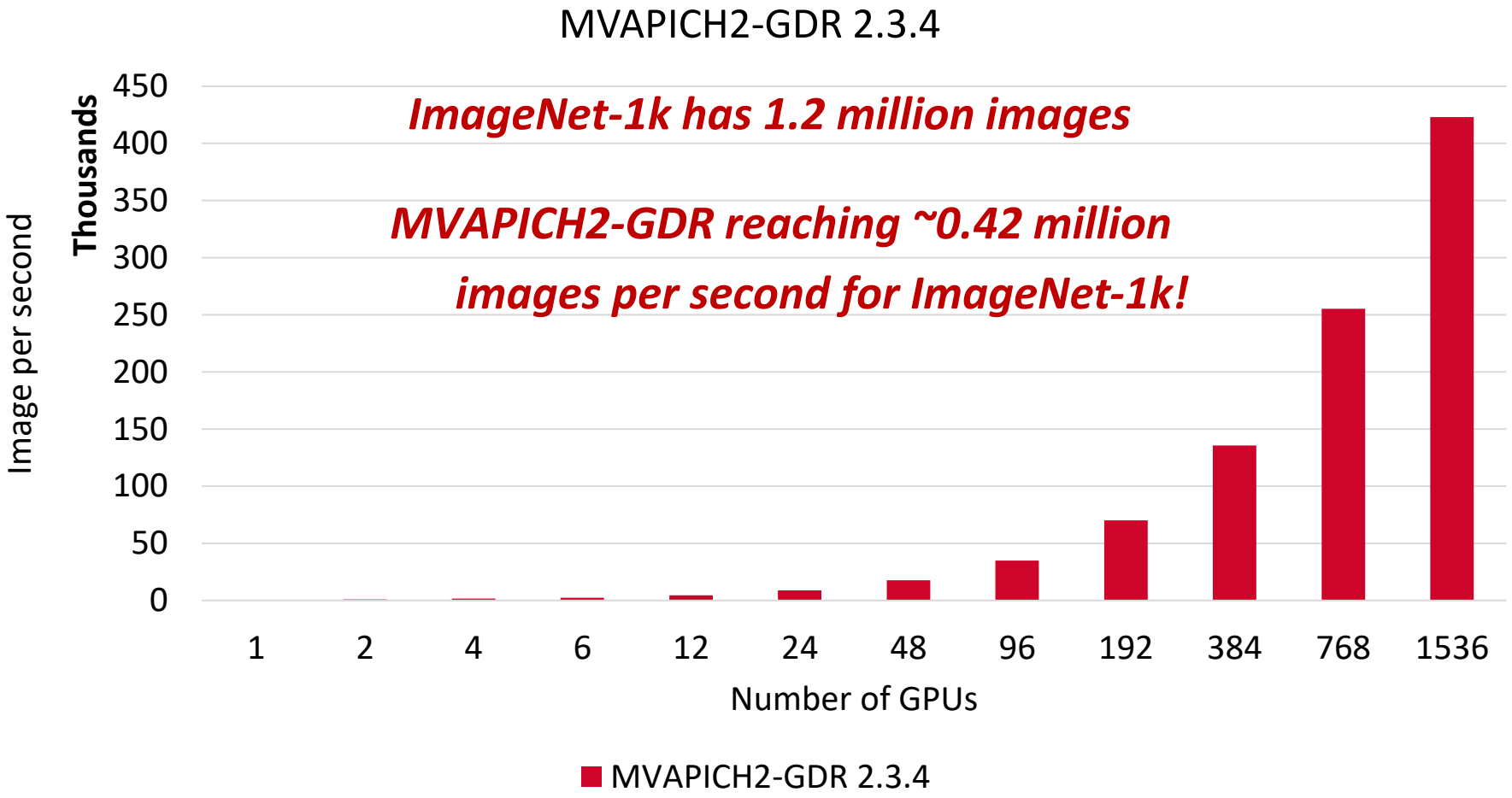
Platform: OSC Pitzer system (8 nodes with 2 Nvidia Volta GPUs), CUDA 11.6





Distributed TensorFlow on ORNL Summit (1,536 GPUs)

- ResNet-50 Training using TensorFlow benchmark on SUMMIT -- 1536 Volta GPUs!
- 1,281,167 (1.2 mil.) images
- Time/epoch = 3 seconds
- Total Time (90 epochs) = 3 x 90 = 270 seconds = **4.5 minutes!**

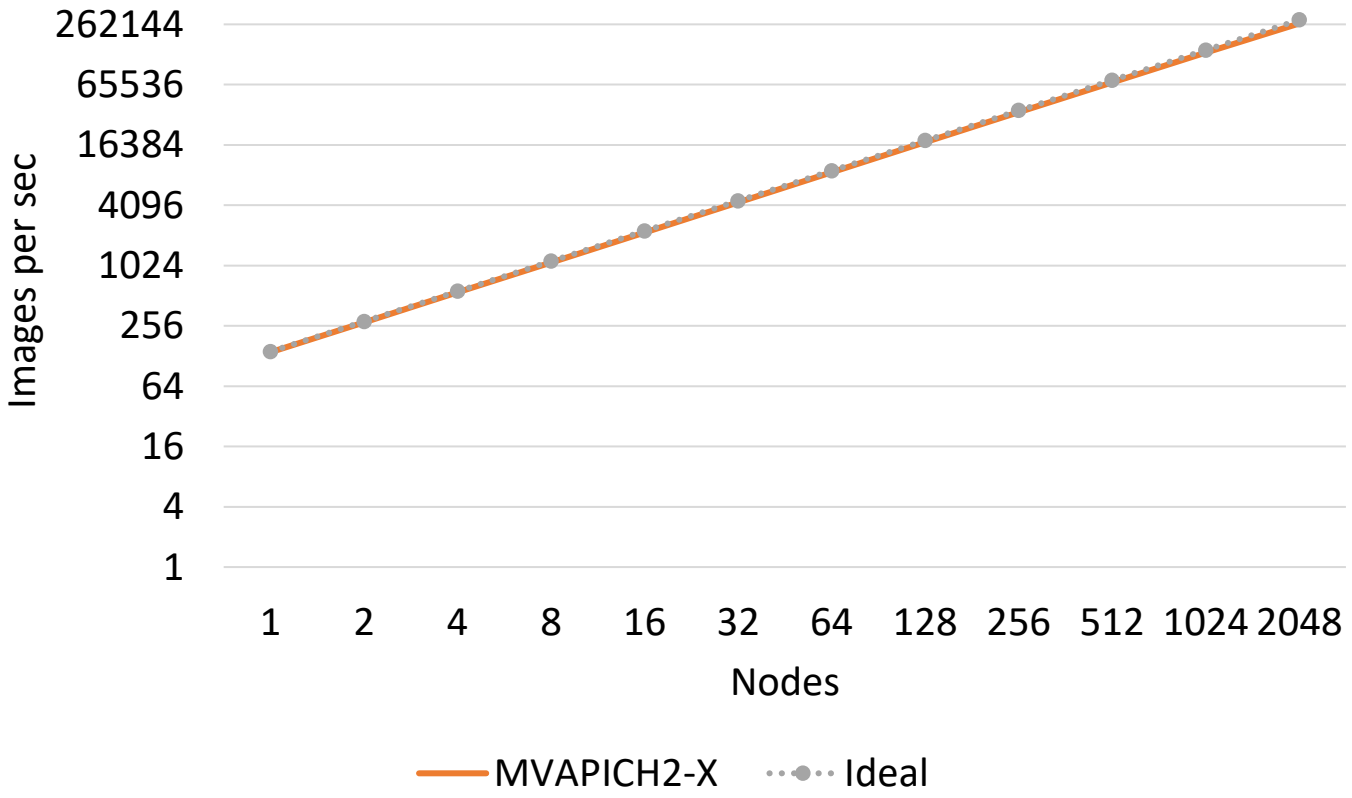


*We observed issues for NCCL2 beyond 384 GPUs

Platform: The Summit Supercomputer (#2 on Top500.org) – 6 NVIDIA Volta GPUs per node connected with NVLink, CUDA 10.1

Distributed TensorFlow on TACC Frontera (2048 CPU nodes)

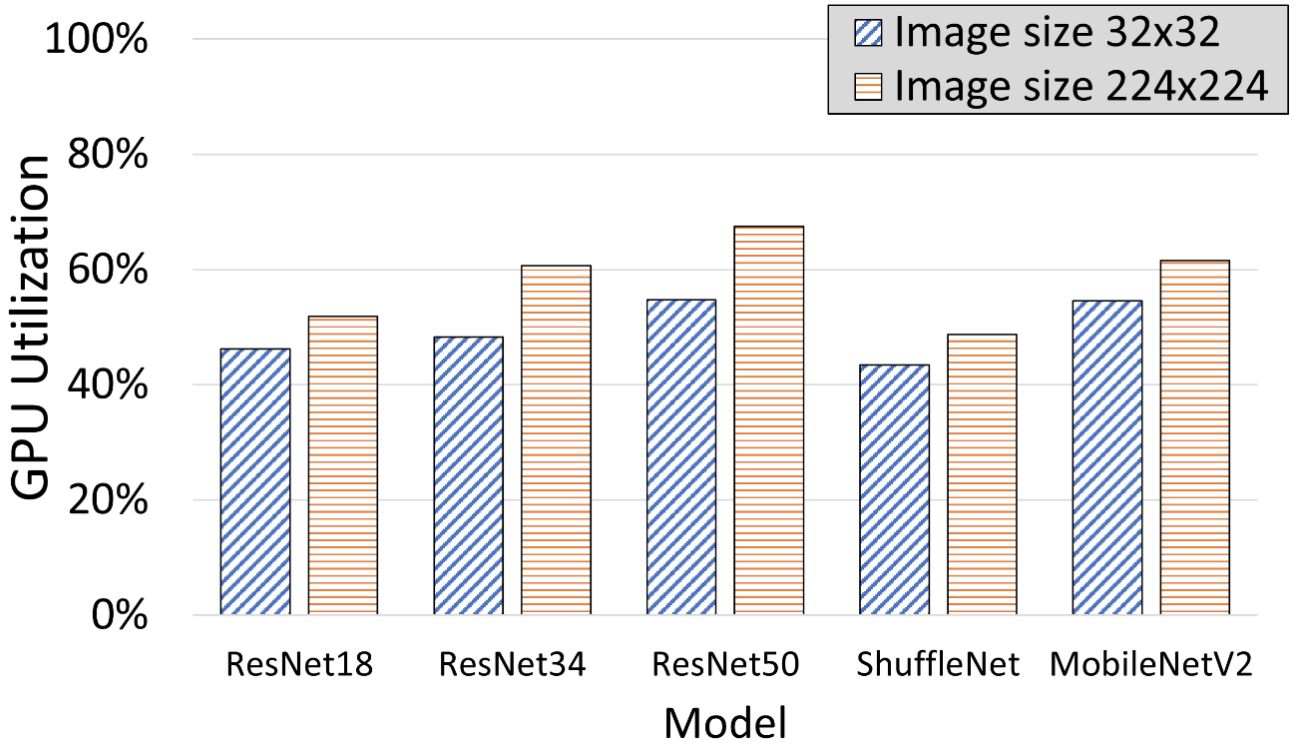
- Scaled TensorFlow to 2048 nodes on Frontera using MVAPICH2 and IntelMPI
- MVAPICH2 delivers close to the ideal performance for DNN training
- Report a peak of 260,000 images/sec on 2048 nodes
- On 2048 nodes, ResNet-50 can be trained in 7 minutes!



A. Jain, A. A. Awan, H. Subramoni, DK Panda, "Scaling TensorFlow, PyTorch, and MXNet using MVAPICH2 for High-Performance Deep Learning on Frontera", DLS '19 (SC '19 Workshop).

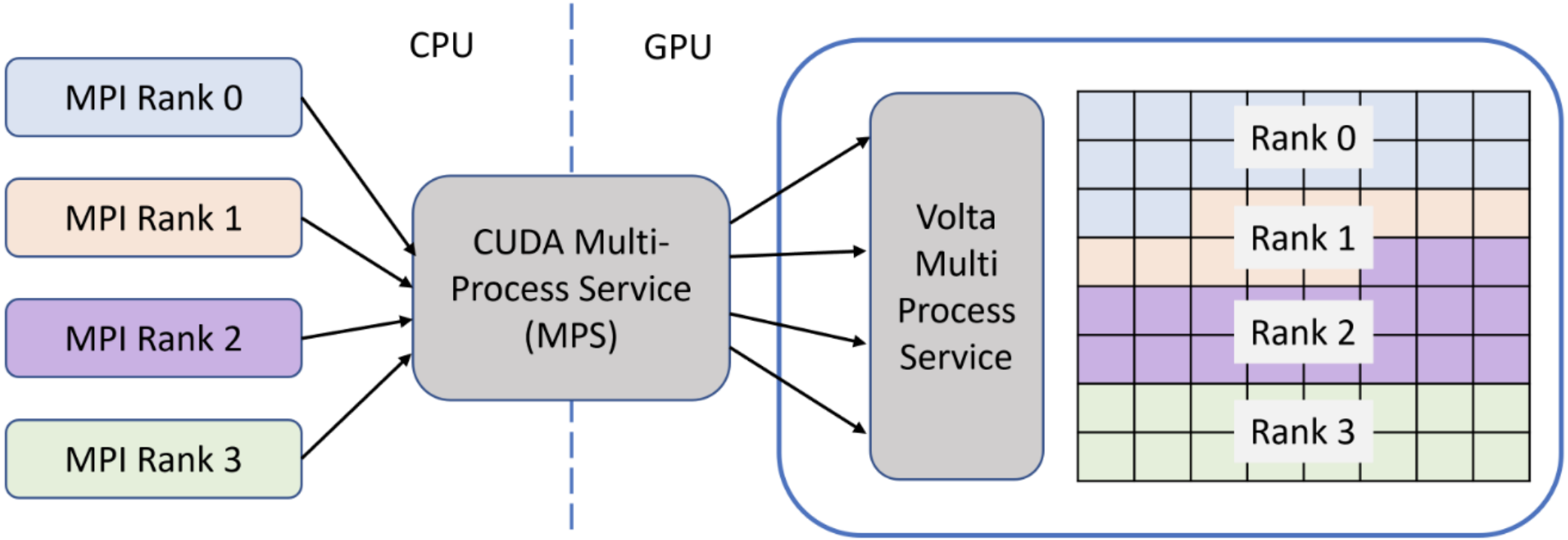
GPU Utilization: Distributed Data-Parallel Training

- Modern GPUs like A100 are very powerful devices
- In some cases, DNN training cannot utilize all compute cores
- GPU utilization can be increased by increasing the batch size
  - There is a limit!
- GPU utilization increases with input image size
- How can we increase GPU utilization?

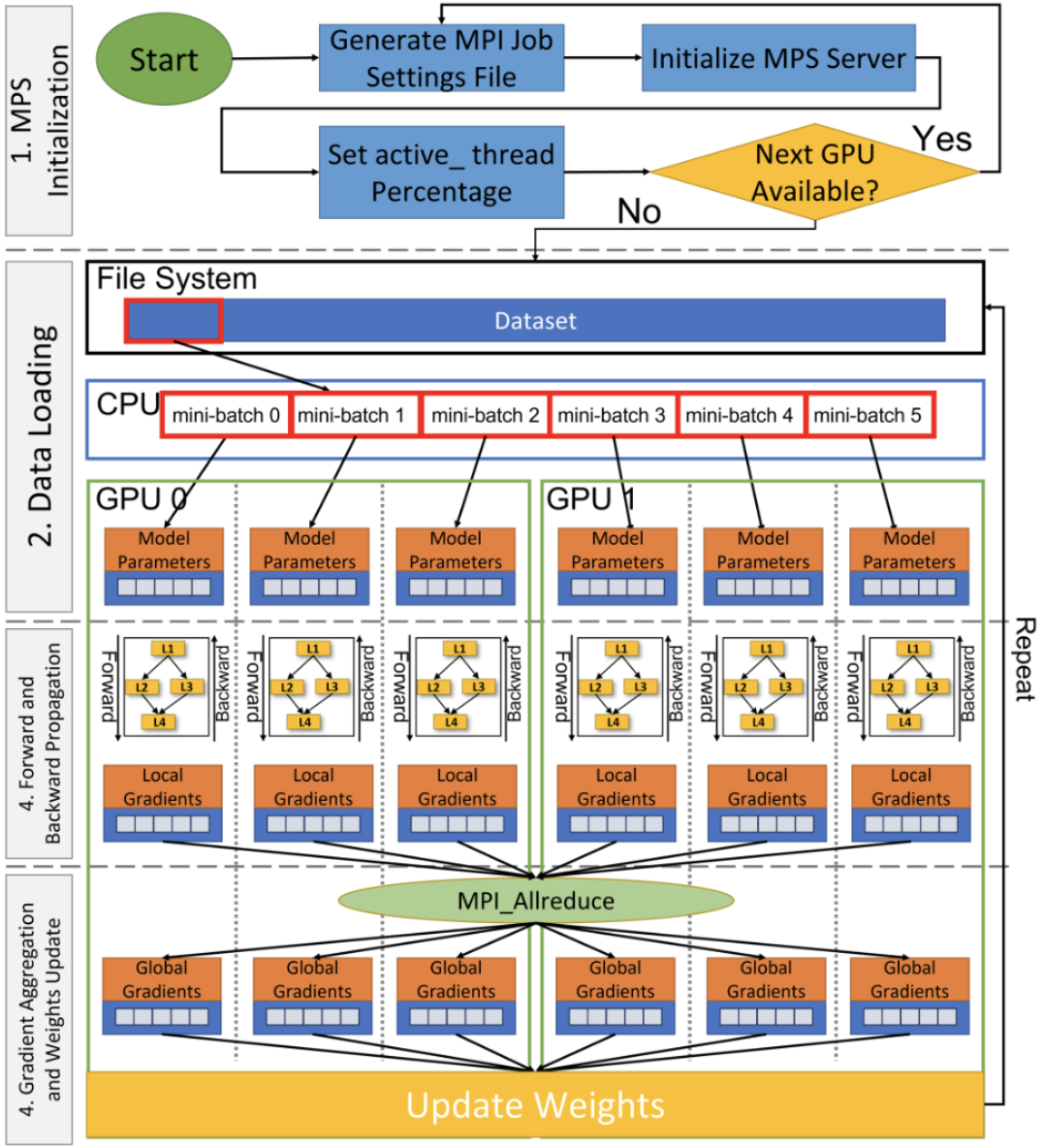


MPS: Multi-Process Service

- MPS is a client-server runtime implementation of the CUDA API that enables the sharing of GPU resources.
  - Logical partitioning

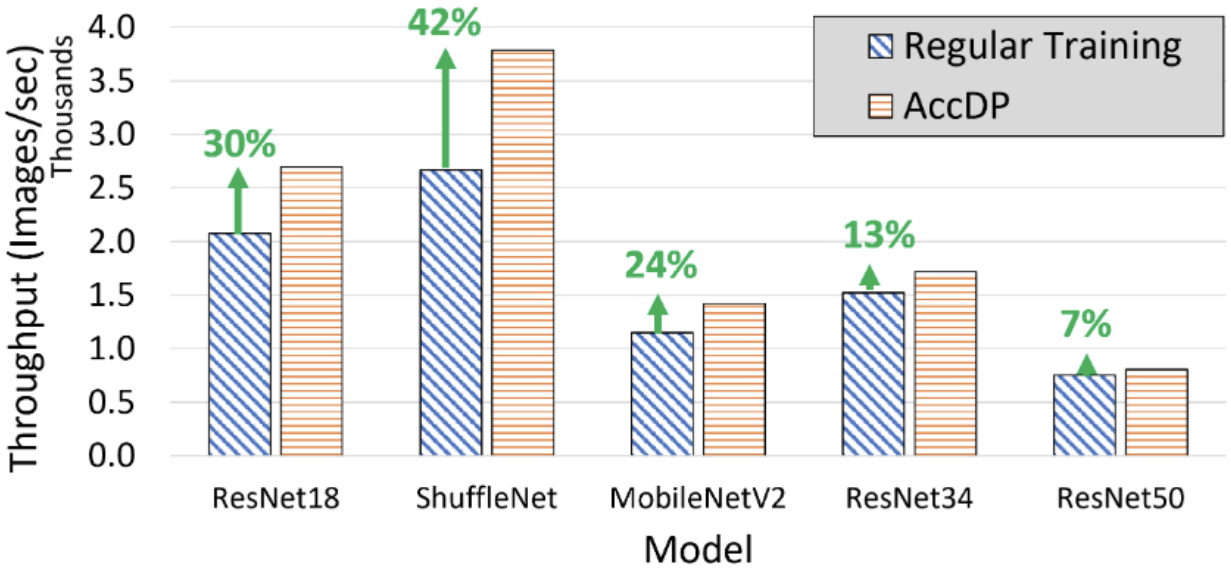


AccDP: Accelerated Data Parallelism using MV2-GDR

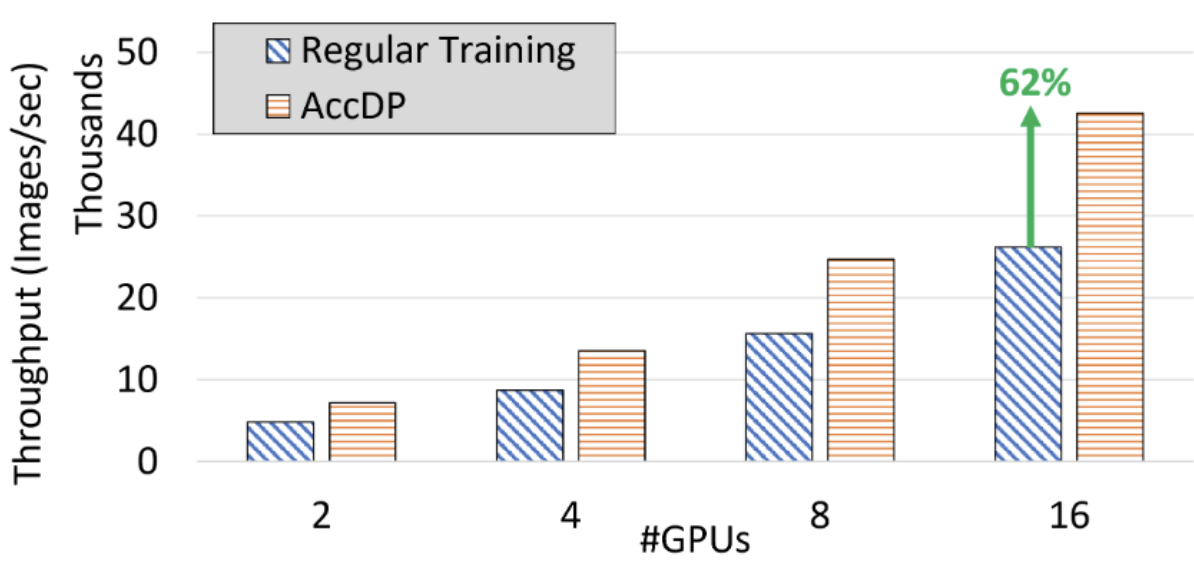


Accelerating Data-Parallel Training using MVAPICH2

- Image Size: 256 X 256
- 4 MPS clients per GPU
- Up to 62% better performance than traditional Data-Parallel training



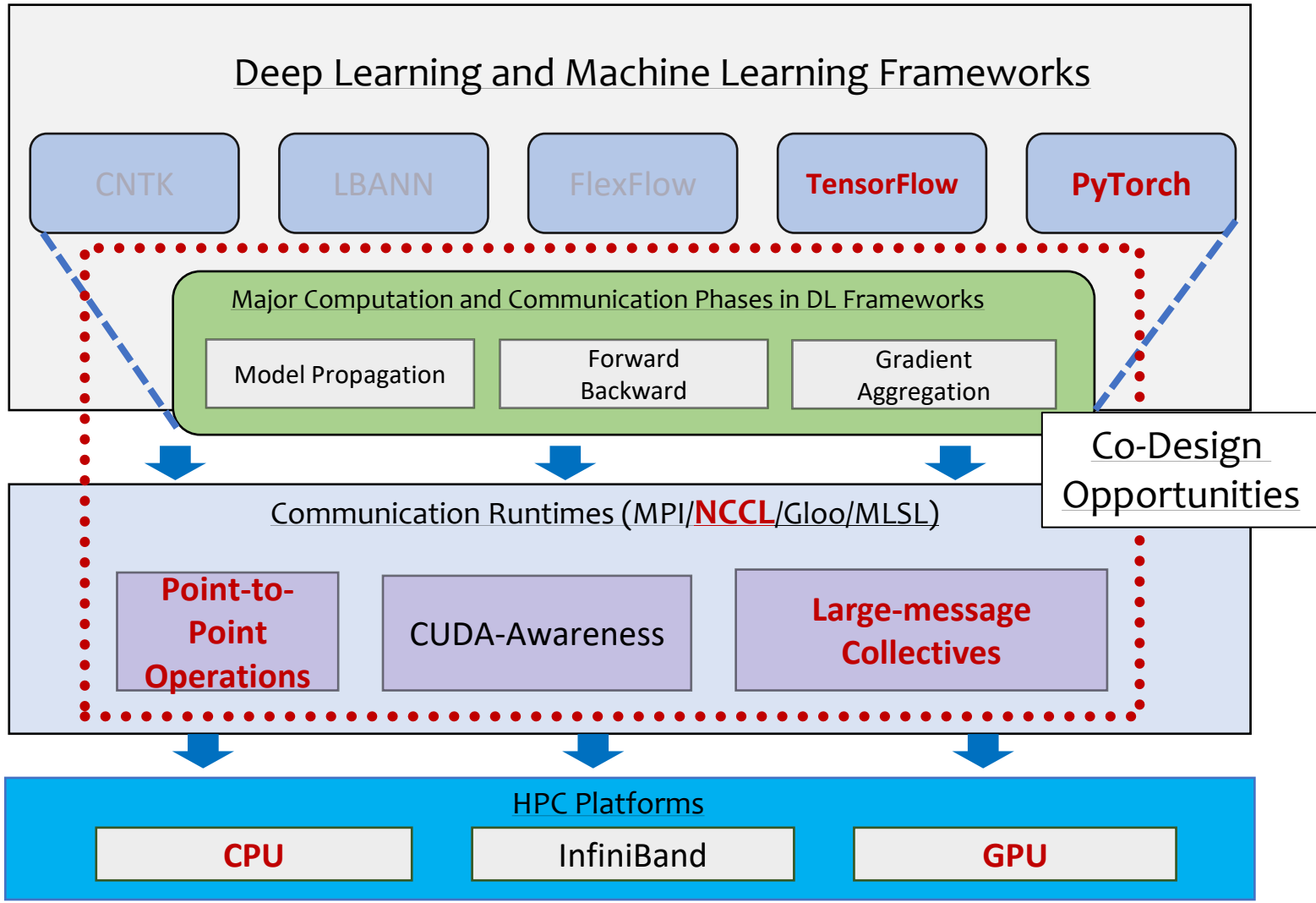
Single Node



Multi Node

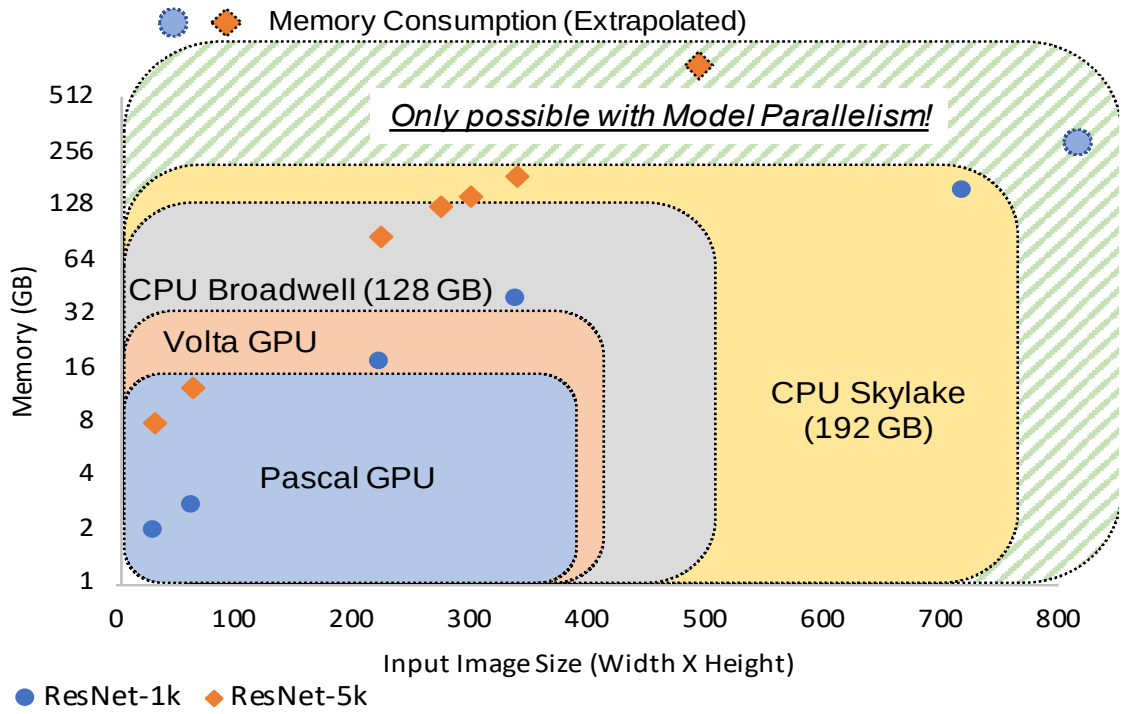
Solutions and Case Studies: Exploiting HPC for DL

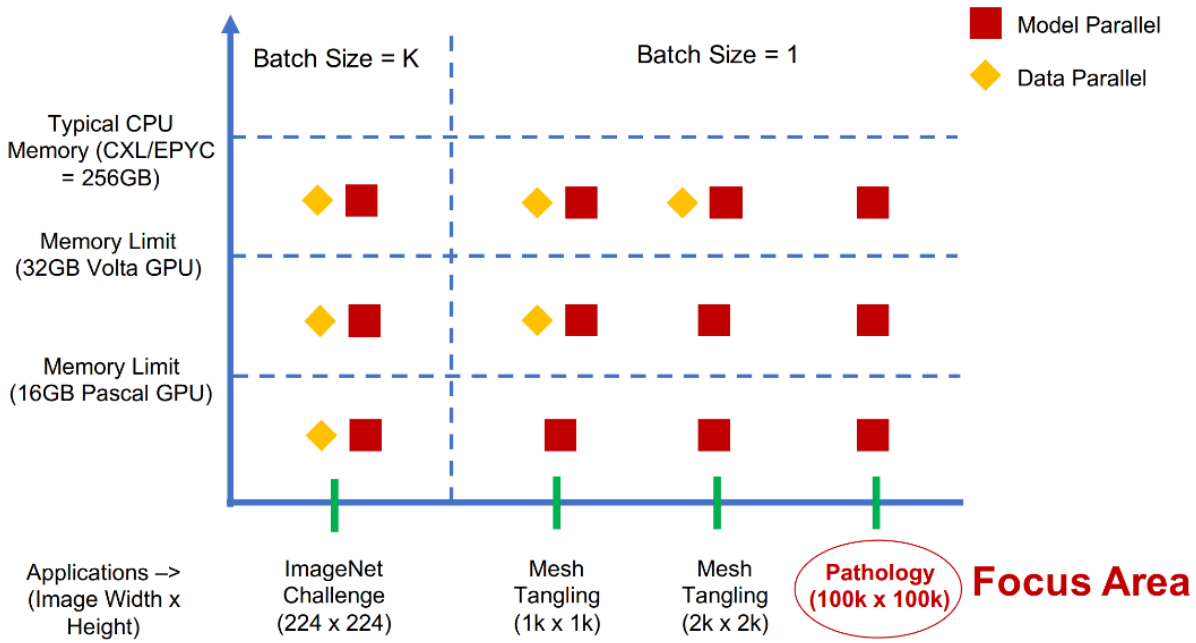
- Data Parallelism
- **Model-Parallelism**



Motivation for Model-Parallelism

- Data-Parallelism– only for models that fit the memory
- Out-of-core models
  - Deeper model → Better accuracy but more memory required!
- Model parallelism can work for out-of-core models!

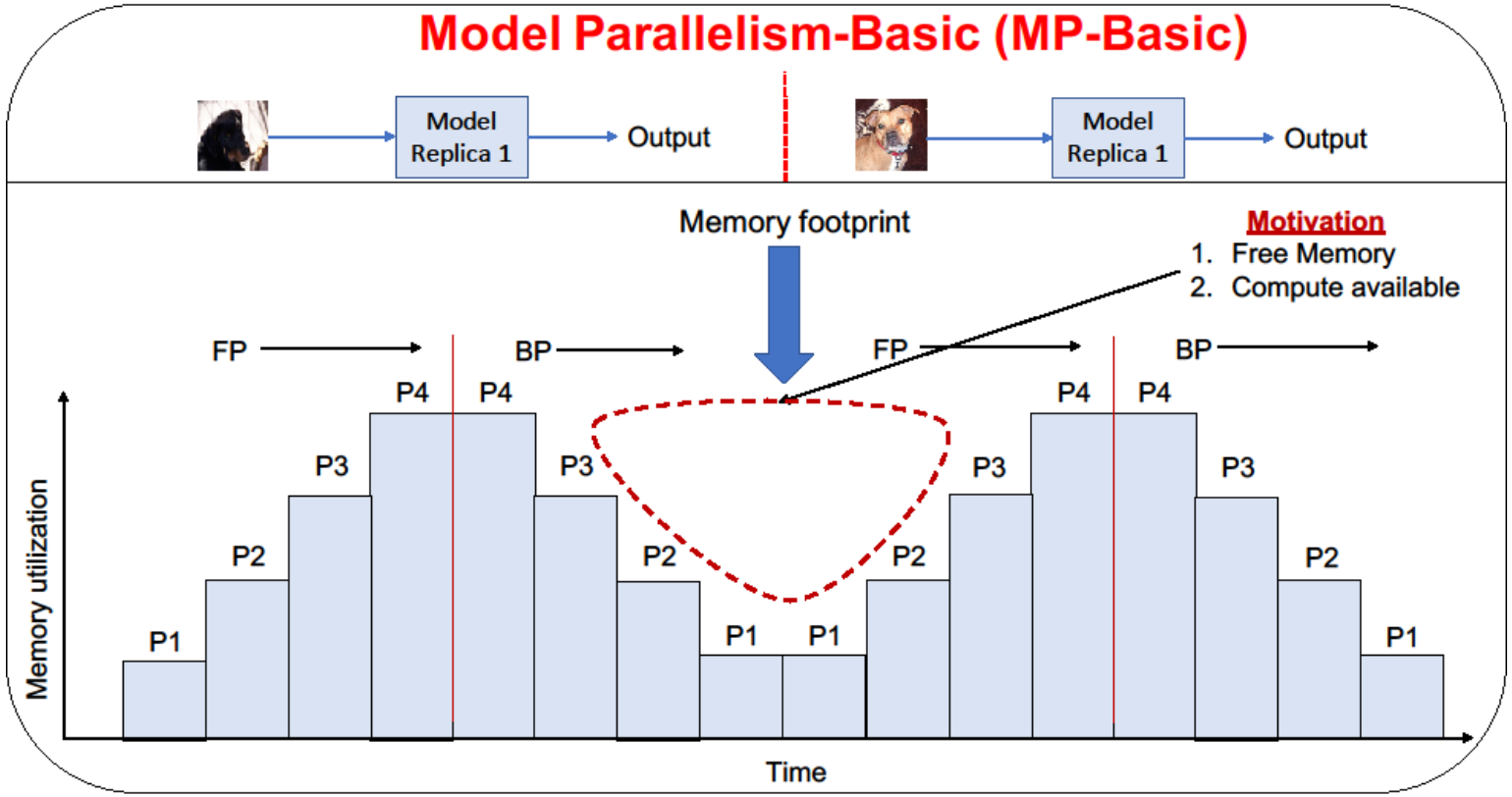




GEMS: GPU Enabled Memory Aware Model Parallelism Systems

Why do we need Memory aware designs?

- Data and Model Parallel training has limitation!
- Maximum Batch Size depends on the memory.
- Basic Model Parallelism suffers from underutilization of memory and compute →

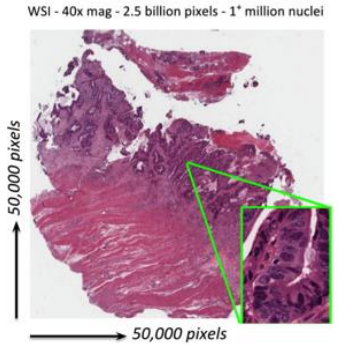


Memory requirement increases with the increase in image size!

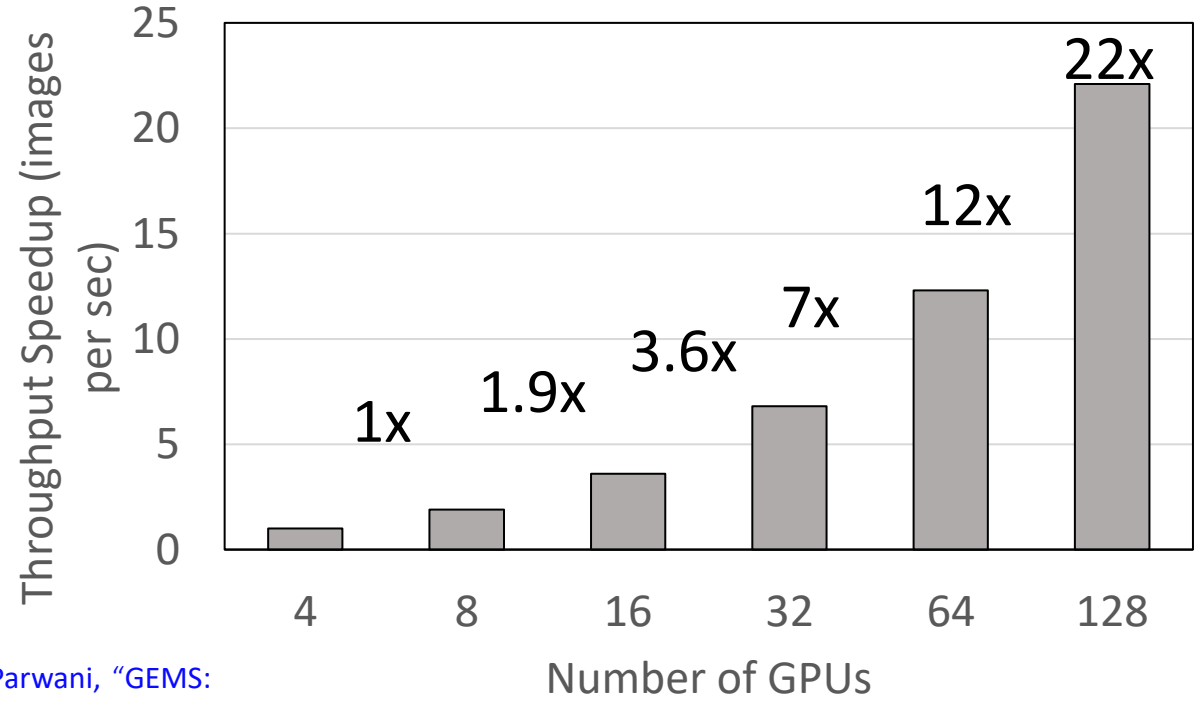
A. Jain, A. Awan, A. Aljuhani, J. Hashmi, Q. Anthony, H. Subramoni, D. Panda, R. Machiraju, A. Parwani, "GEMS: GPU Enabled Memory Aware Model Parallelism System for Distributed DNN", SC '20.

Exploiting Model Parallelism in AI-Driven Digital Pathology

- Pathology whole slide image (WSI)
  - Each WSI = 100,000 x 100,000 pixels
  - Can not fit in a single GPU memory
  - Tiles are extracted to make training possible
- Two main problems with tiles
  - Restricted tile size because of GPU memory limitation
  - Smaller tiles loose structural information
- Reduced training time significantly
  - **GEMS-Basic: 7.25 hours (1 node, 4 GPUs)**
  - **GEMS-MAST: 6.28 hours (1 node, 4 GPUs)**
  - **GEMS-MASTER: 4.21 hours (1 node, 4 GPUs)**
  - **GEMS-Hybrid: 27 mins (32 nodes, 128 GPUs)**
  - **Overall 15x reduction in training time!!!!**



Courtesy: https://blog.kitware.com/digital-slide-archive-large-image-and-histomicstk-open-source-informatics-tools-for-management-visualization-and-analysis-of-digital-histopathology-data/

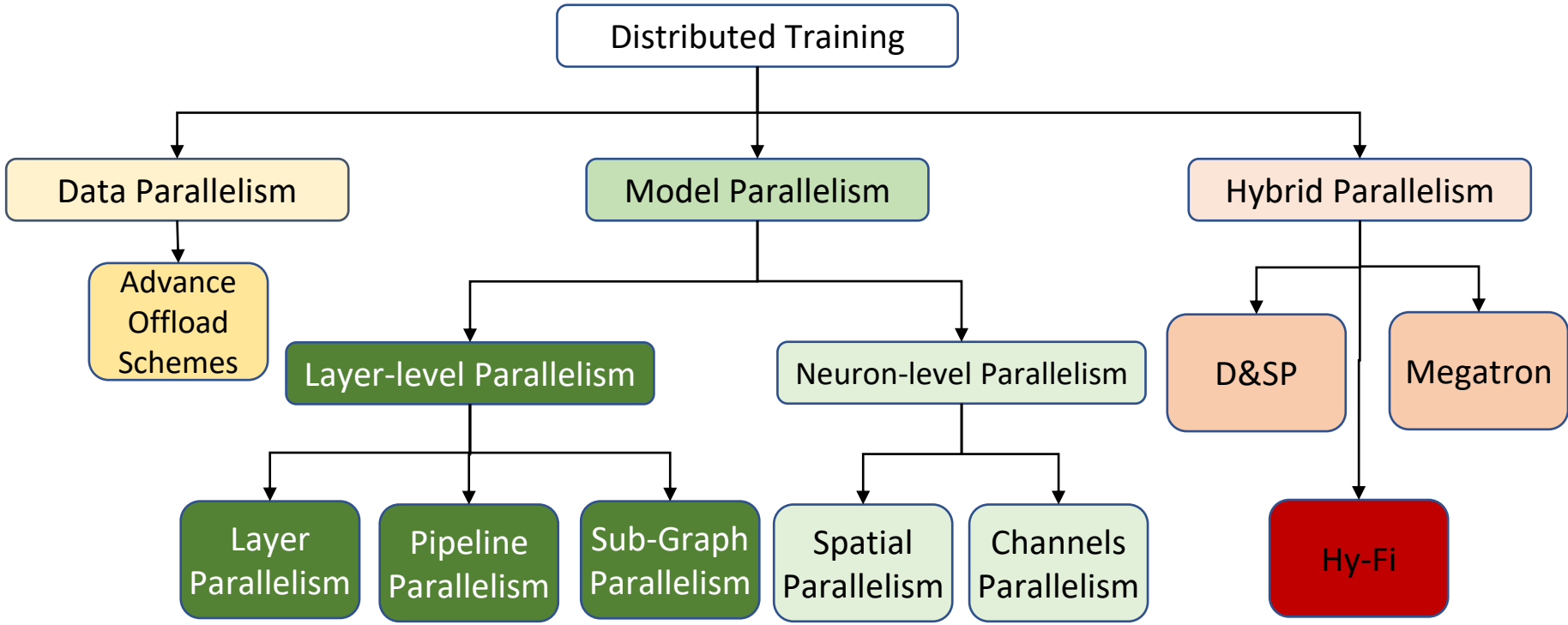


Scaling ResNet110 v2 on 1024×1024 image tiles using histopathology data

A. Jain, A. Awan, A. Aljuhani, J. Hashmi, Q. Anthony, H. Subramoni, D. K. Panda, R. Machiraju, and A. Parwani, "GEMS: GPU Enabled Memory Aware Model Parallelism System for Distributed DNN Training", Supercomputing (SC '20).

Parallelization Strategies

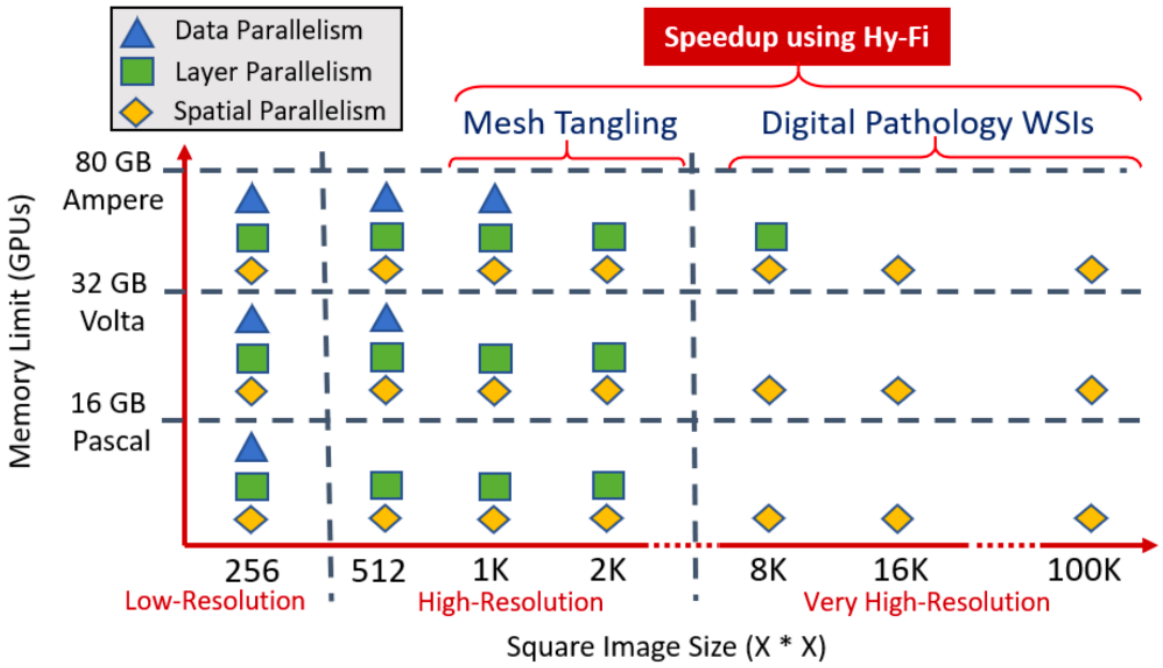
- Data Parallelism
- Model Parallelism
  - Layer-level Parallelism
    - Layer
    - Pipeline
    - Sub-Graph
  - Neuron-level Parallelism
    - Spatial
    - Channel
- Hybrid Parallelism
  - D&SP
  - Megatron
  - **Hy-Fi**



Jain, Arpan, et al. "Hy-Fi: Hybrid Five Dimensional Parallel DNN Training on High-Performance GPU Clusters." *International Conference on High Performance Computing*. Springer, Cham, 2022.

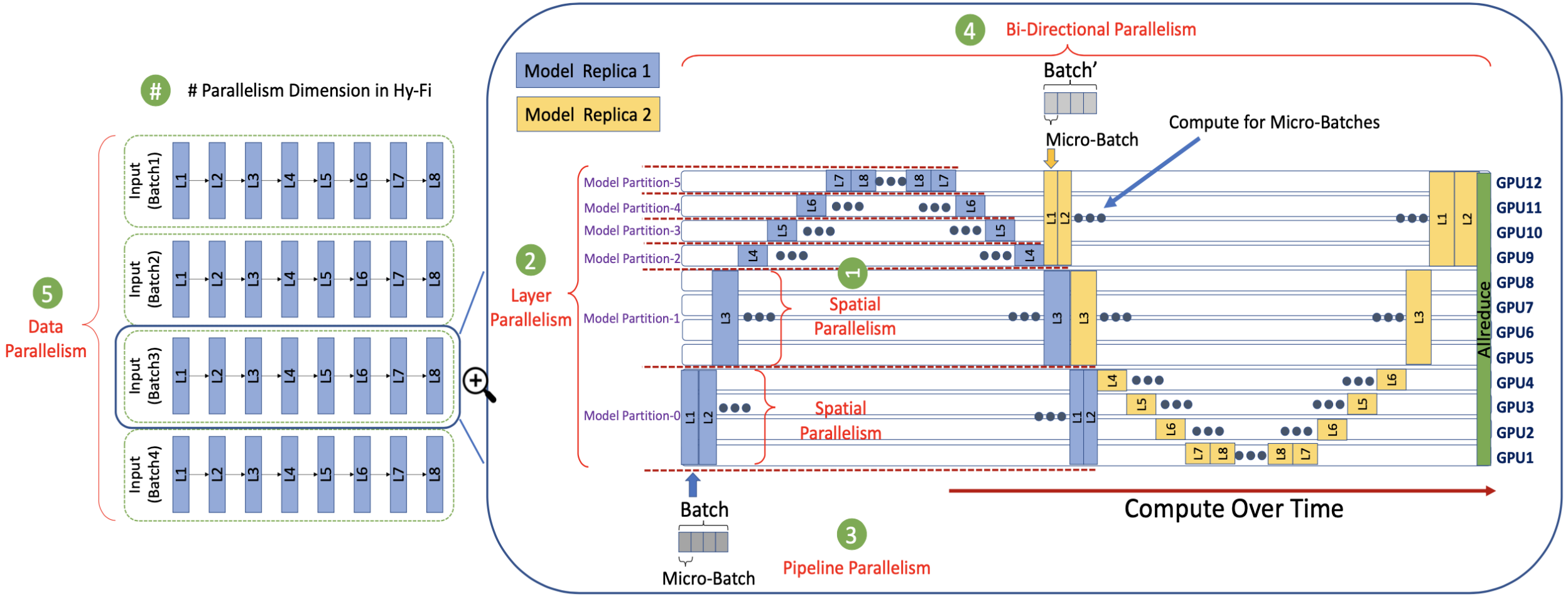
Why Hybrid Parallelism?

- Data-Parallelism– only for models that fit the memory
- Out-of-core models
  - Deeper model → Better accuracy but more memor required!
  - Real-world applications have very high-resolution images →
- Model parallelism can work for out-of-co models!
  - Performance is questionable!
  - Layer-parallelism is not enough



Proposed Hybrid Five-Dimensional Parallelism (Hy-Fi)

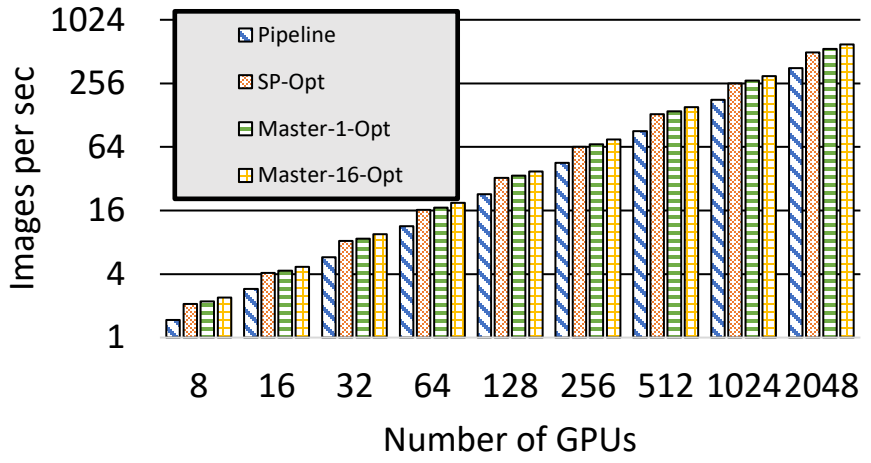
- Integrates spatial, layer, pipeline, bi-directional, and data parallelism



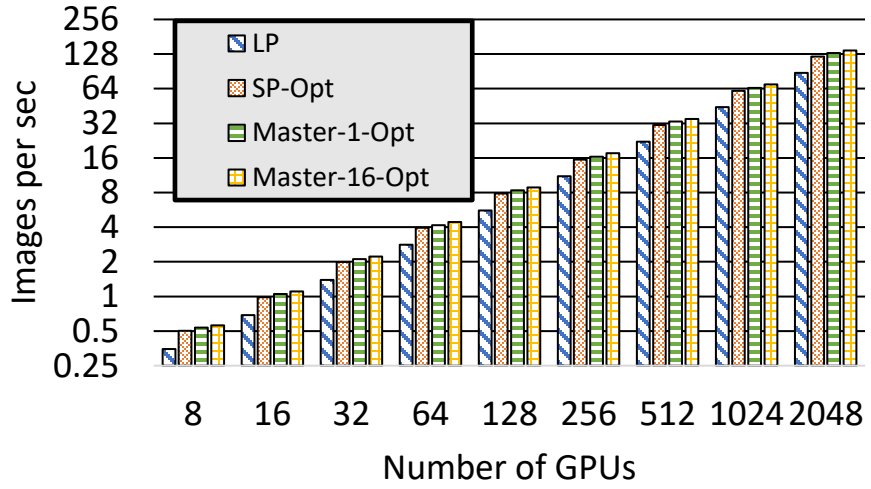
Jain, Arpan, et al. "Hy-Fi: Hybrid Five Dimensional Parallel DNN Training on High-Performance GPU Clusters." *International Conference on High Performance Computing*. Springer, Cham, 2022.

Accelerating Out-of-core Training at Scale

- Approach
  - LP: Layer Parallelism
  - Pipeline: Pipeline Parallelism
  - SP: Spatial Parallelism
  - Master: Hy-Fi
  - Opt: with Communication optimization in Hy-Fi
- Setup
  - Image Size: 2048 X 2048
  - 2048 NVIDIA V100 GPUs
- Speedup
  - Up to **2.67X** over Layer Parallelism (LP)
  - Near-linear scaling (**94.5%**) on 2,048 GPUs



AmobeNet-f214

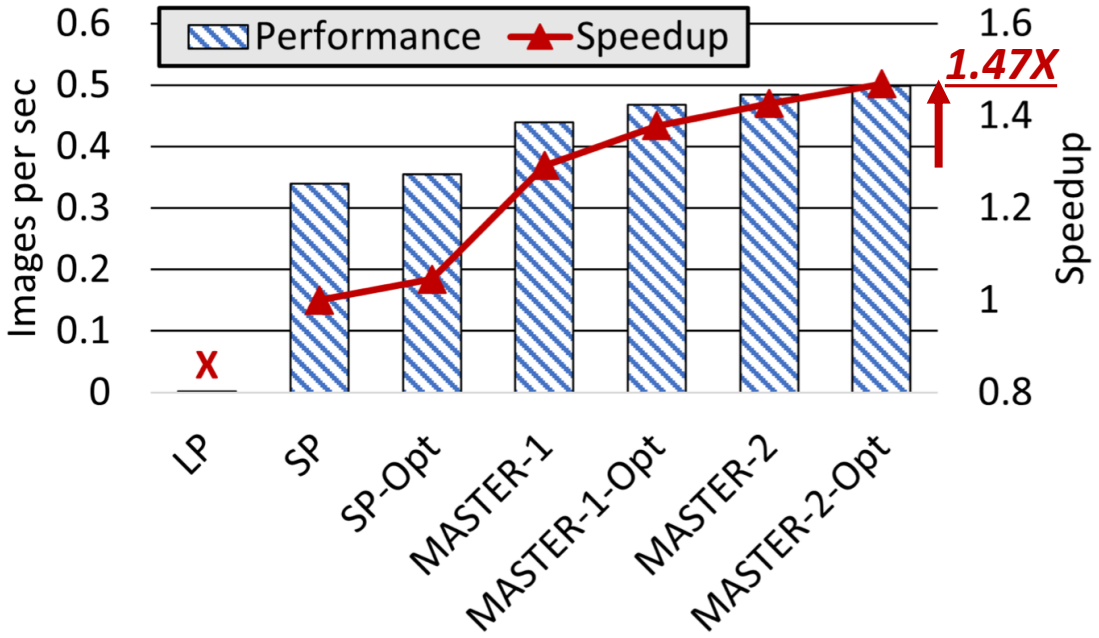


AmobeNet-f416

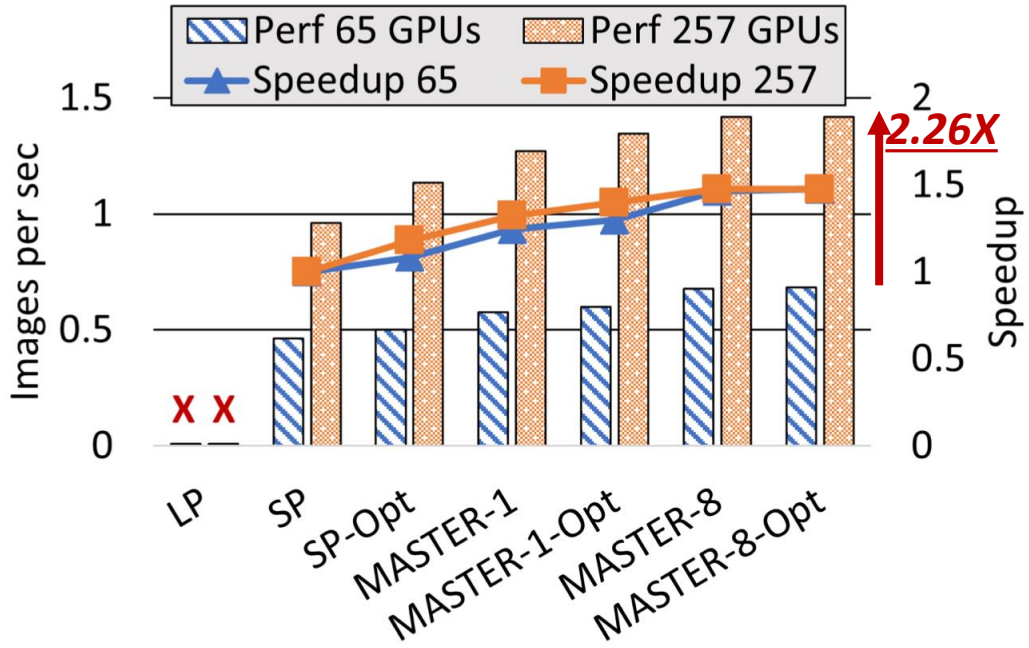
Jain, Arpan, et al. "Hy-Fi: Hybrid Five Dimensional Parallel DNN Training on High-Performance GPU Clusters." *International Conference on High Performance Computing*. Springer, Cham, 2022.

Enabling Training on Very High-Resolution Images

- Enabled training on 8,192 X 8,192 and 16,384 X 16,384 images sizes
- Speedup over basic spatial parallelism
  - 8,192 X 8,192 Images: 1.476X and 2.26X (Strong Scaling)
  - 16,384 X 16,384 Images: 1.47X



AmobeNet-f214 on 16,384 X 16,384 images



AmobeNet-f214 on 8,192 X 8,192 images

Outline

- Introduction
- Deep Learning
  - Deep Neural Networks
  - Distributed Deep Learning
  - DL Solutions
- Conclusion

Conclusion

- Exponential growth in Machine Learning/Deep Learning/Data Science frameworks
- Provided an overview of issues, challenges, and opportunities for designing efficient communication runtimes
  - Efficient, scalable, and hierarchical designs are crucial for ML/DL/Data Science frameworks
  - Co-design of communication runtimes and ML/DL/Data Science frameworks will be essential
- Presented use-cases to demonstrate the complex interaction between DL middleware with the underling HPC technologies and middleware
- Need collaborative efforts to achieve the full potential

Funding Acknowledgments

Funding Support by

Equipment Support by

Acknowledgments to all the Heroes (Past/Current Students and Staffs)

Current Students (Graduate)

- N. Alnaasan (Ph.D.)
- Q. Anthony (Ph.D.)
- C.-C. Chun (Ph.D.)
- N. Contini (Ph.D.)
- A. Jain (Ph.D.)
- K. S. Khorassani (Ph.D.)
- P. Kousha (Ph.D.)
- B. Michalowicz (Ph.D.)
- B. Ramesh (Ph.D.)
- K. K. Suresh (Ph.D.)
- A. H. Tu (Ph.D.)
- S. Xu (Ph.D.)
- Q. Zhou (Ph.D.)
- K. Al Attar (M.S.)
- L. Xu (Ph.D.)
- H. Ahn (Ph.D.)
- G. Kuncham (Ph.D.)
- R. Vaidya (Ph.D.)
- J. Yao (P.hD.)
- M. Han (M.S.)
- A. Guptha (M.S.)

Current Research Scientists

- M. Abduljabbar
- A. Shafi

Current Students (Undergrads)

- V. Shah
- T. Chen

Current Faculty

- H. Subramoni

Current Software Engineers

- B. Seeds
- N. Pavuk
- N. Shineman
- M. Lieber

Current Research Specialist

- R. Motlagh

Past Students

- A. Awan (Ph.D.)
- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- M. Bayatpour (Ph.D.)
- R. Biswas (M.S.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- S. Chakraborthy (Ph.D.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- C.-H. Chu (Ph.D.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- J. Hashmi (Ph.D.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- M. Kedia (M.S.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- K. Kulkarni (M.S.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- M. Li (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- K. Raj (M.S.)
- R. Rajachandrasekar (Ph.D.)
- D. Shankar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- N. Sarkauskas (B.S. and M.S)
- N. Senthil Kumar (M.S.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Srivastava (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)
- J. Zhang (Ph.D.)

Past Research Scientists

- K. Hamidouche
- S. Sur
- X. Lu

Past Senior Research Associate

- J. Hashmi

Past Programmers

- A. Reifsteck
- D. Bureddy
- J. Perkins

Past Post-Docs

- D. Banerjee
- X. Besseron
- M. S. Ghazimeersaeed
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- K. Manian
- S. Marcarelli
- A. Ruhela
- J. Vienne
- H. Wang

Past Research Specialist

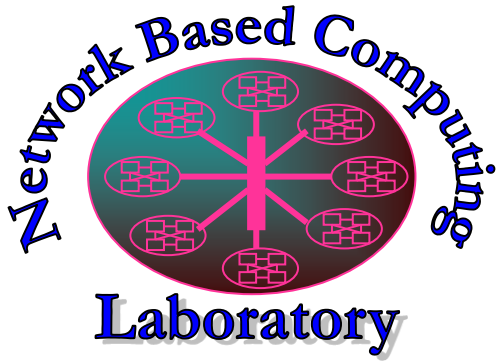
- M. Arnold
- J. Smith

Thank You!

jain.575@osu.edu

Follow us on

https://twitter.com/mvapich



Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/