MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

HiDL
High-Performance Deep Learning

Latest version of the slides can be obtained from

http://www.cse.ohio-state.edu/~panda/s9501.pdf

High Performance Distributed Deep Learning: A Beginner's Guide

Tutorial at GTC '19

by

Dhabaleswar K. (DK) Panda

The Ohio State University

E-mail: panda@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~panda

Ammar Ahmad Awan

The Ohio State University

E-mail: awan.10@osu.edu

http://www.cse.ohio-state.edu/~awan.10

Hari Subramoni

The Ohio State University

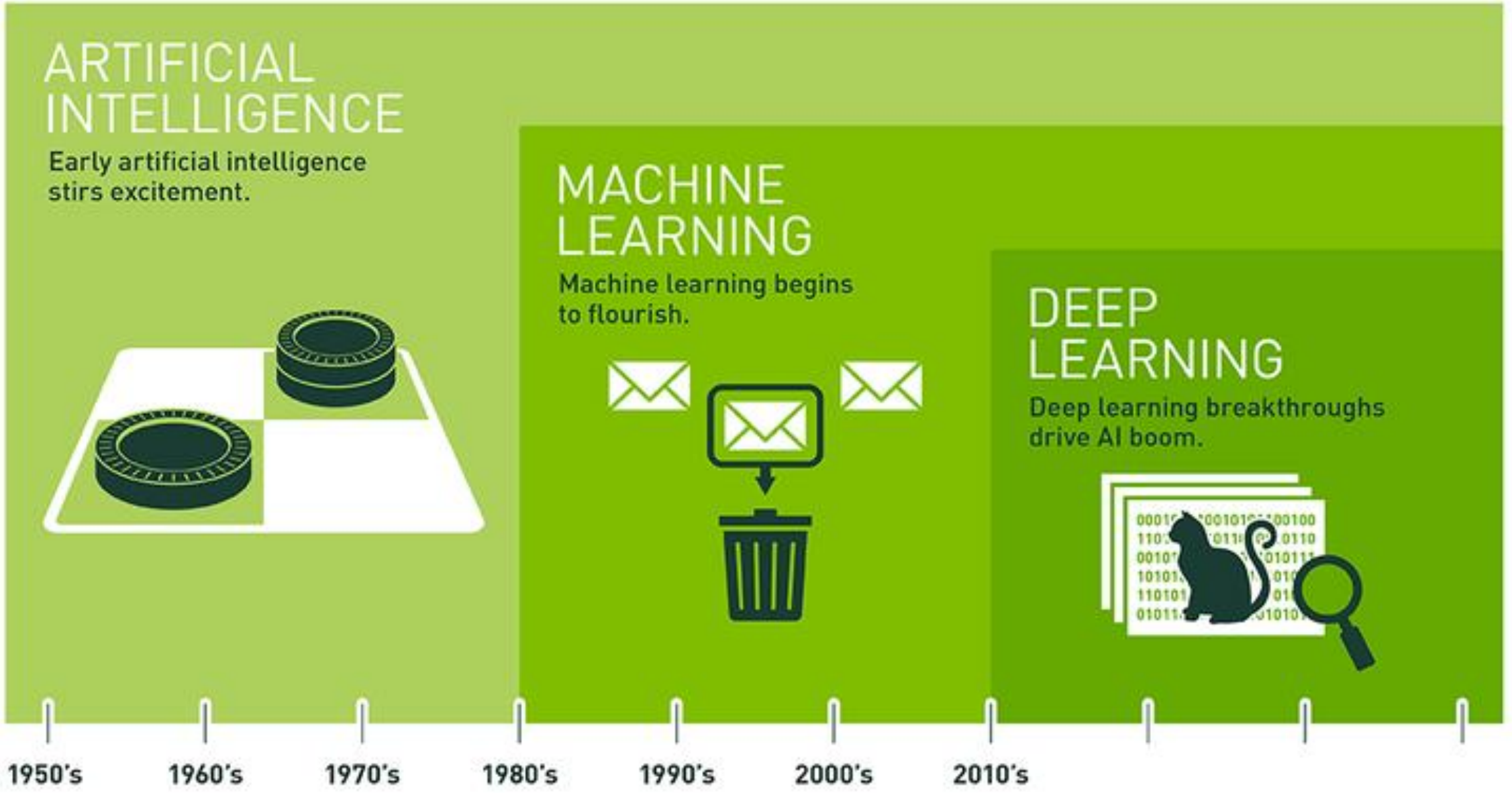
E-mail: subramon@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~subramon

Outline

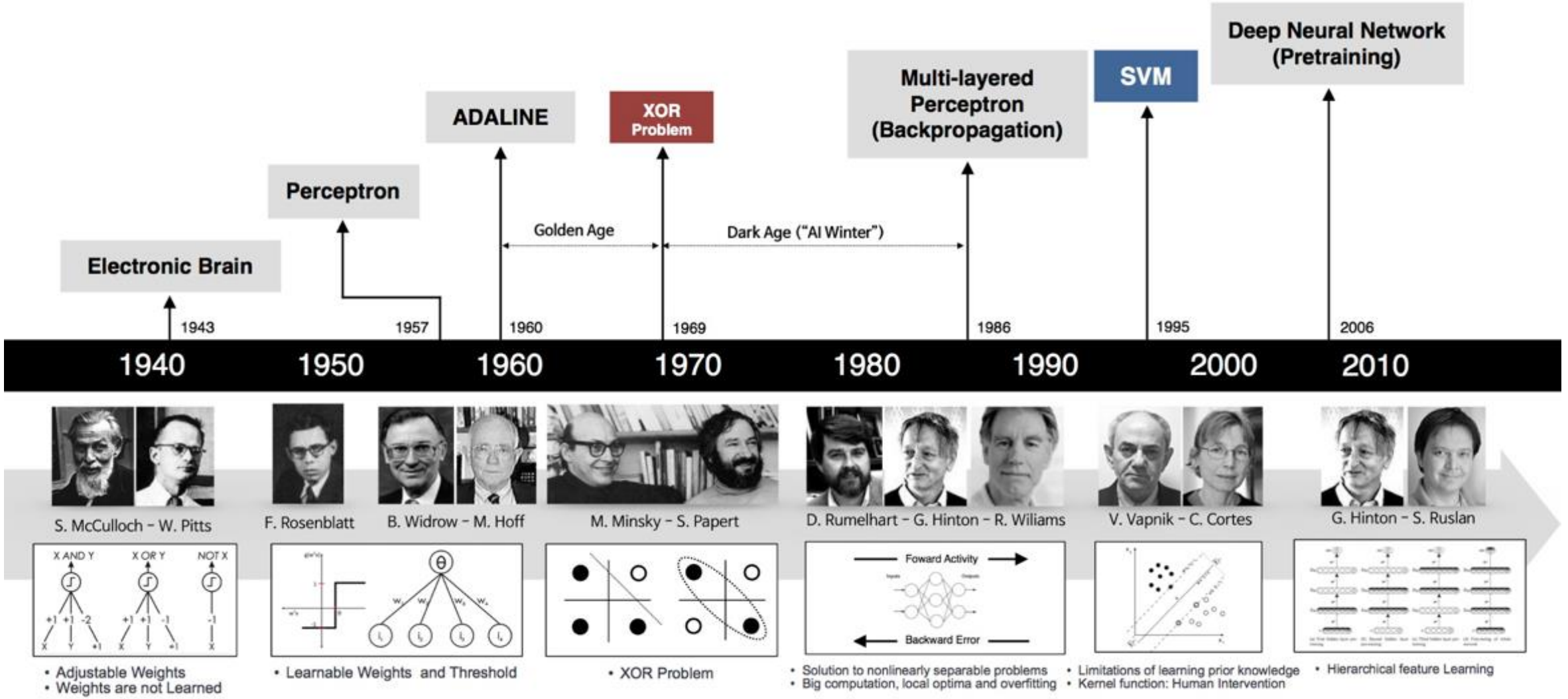
- **Introduction**
  - **The Past, Present, and Future of Deep Learning**
  - What are Deep Neural Networks?
  - Diverse Applications of Deep Learning
  - Deep Learning Frameworks
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Conclusion

Brief History of Deep Learning (DL)



Courtesy: http://www.zdnet.com/article/caffe2-deep-learning-wide-ambitions-flexibility-scalability-and-advocacy/

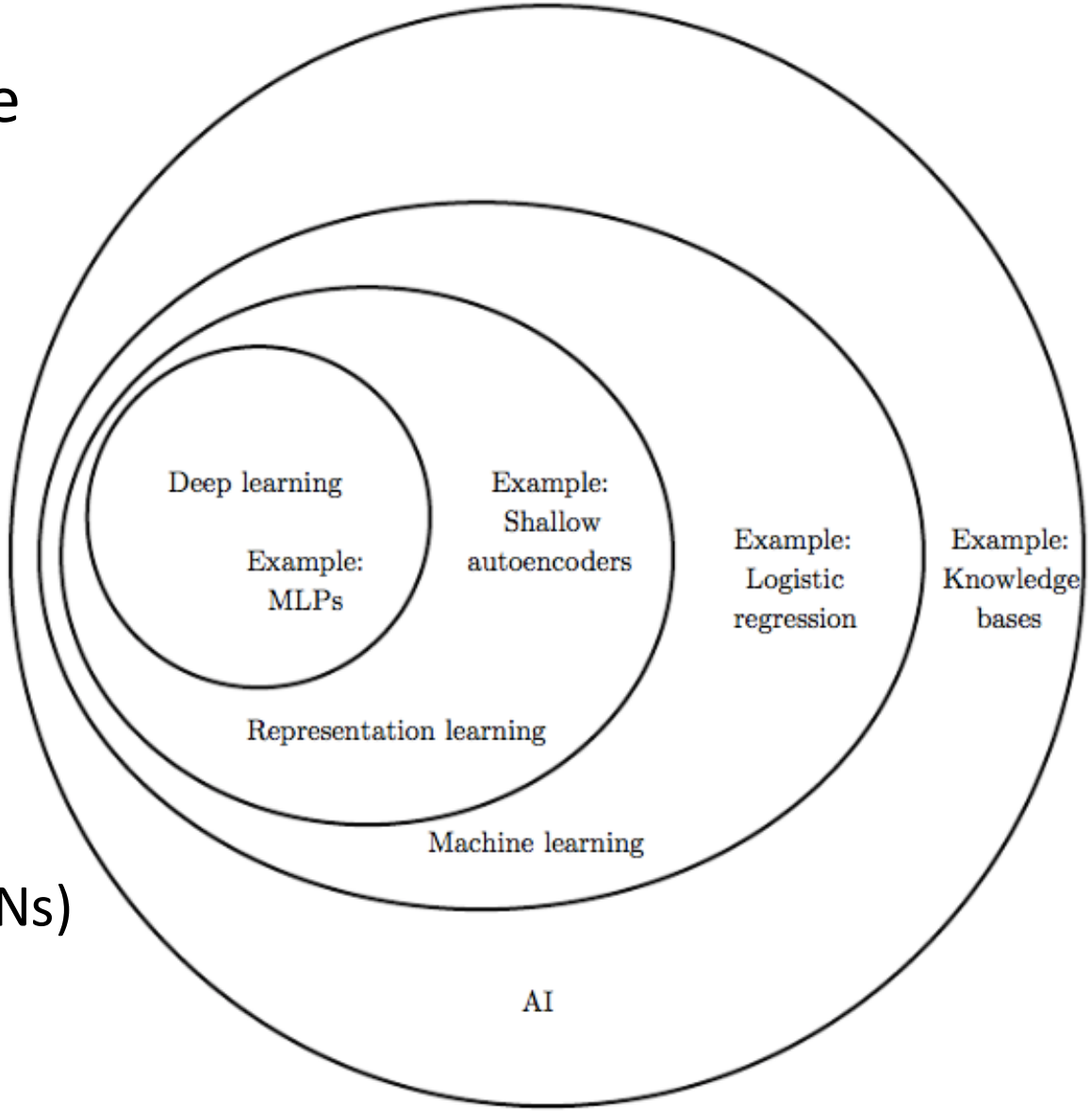
Milestones in the Development of Neural Networks



Courtesy: https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html

Understanding the Deep Learning Resurgence

- Deep Learning is a sub-set of Machine Learning
  - But, it is perhaps the most radical and revolutionary subset
  - Automatic feature extraction vs. hand-crafted features
- Deep Learning
  - A renewed interest and a lot of hype!
  - Key success: Deep Neural Networks (DNNs)
  - Everything was there since the late 80s except the "**computability of DNNs**"

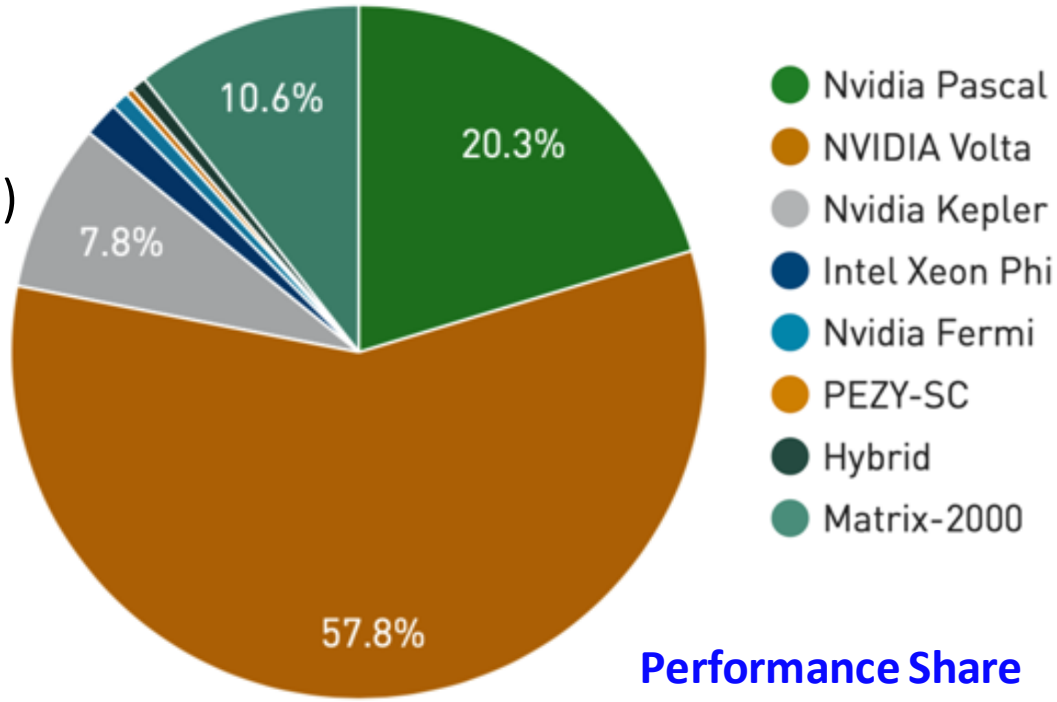


Courtesy: http://www.deeplearningbook.org/contents/intro.html

Deep Learning, Many-cores, and HPC

- NVIDIA GPUs are the main driving force for faster training of DL models
  - The ImageNet Challenge - (ILSVRC)
  - 90% of the ImageNet teams used GPUs in 2014*
  - Deep Neural Networks (DNNs) like AlexNet, GoogLeNet, and VGG are used
  - A natural fit for DL due to the throughput-oriented nature

- In the High Performance Computing (HPC) arena
  - 126/500 Top HPC systems use NVIDIA GPUs (Nov '18)
  - CUDA-Aware Message Passing Interface (MPI)
  - NVIDIA Fermi, Kepler, and Pascal architecture
  - DGX-1 (Pascal) and DGX-2 (Volta)
    - Dedicated DL supercomputers

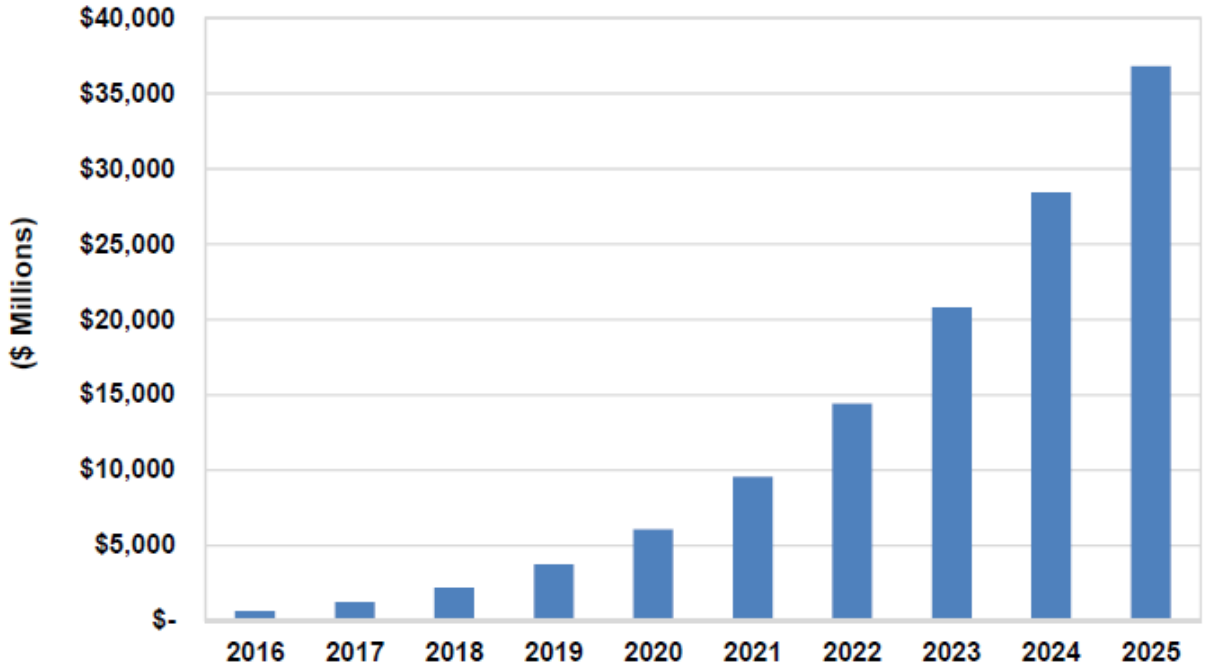


Performance Share
www.top500.org

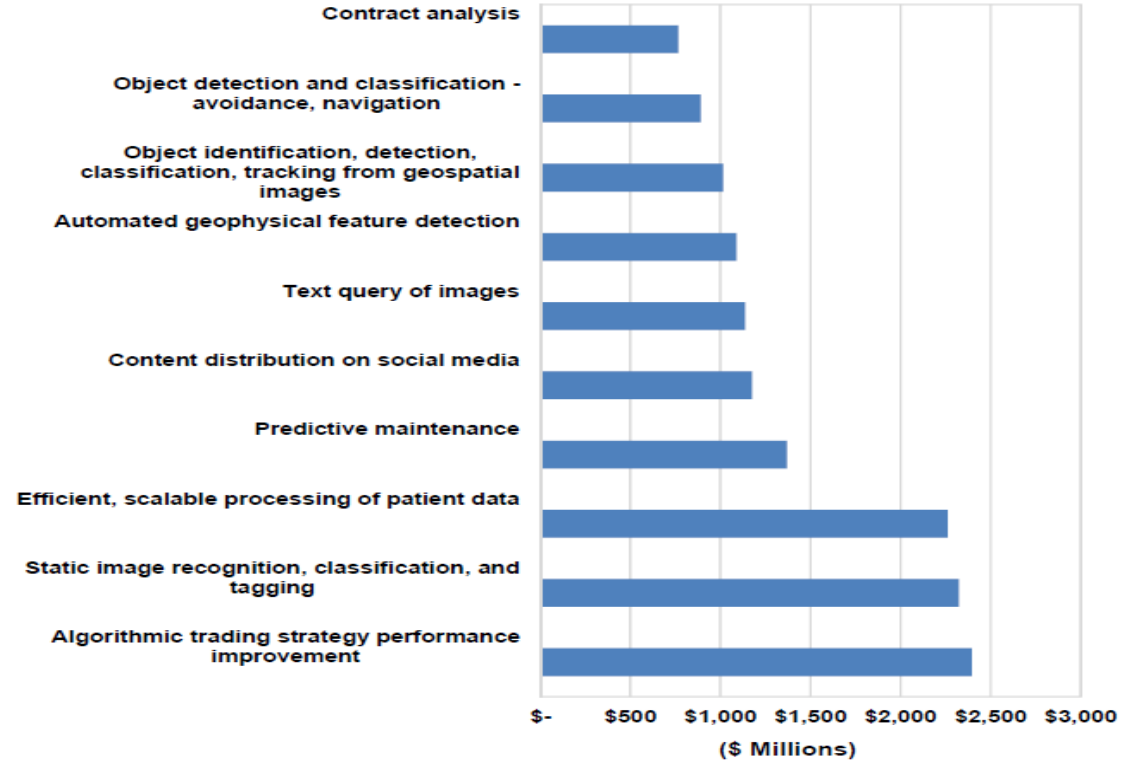
*https://blogs.nvidia.com/blog/2014/09/07/imagenet/

Deep Learning Use Cases and Growth Trends

1.1 *Artificial Intelligence Revenue, World Markets: 2016-2025*



1.2 *Artificial Intelligence Revenue, Top 10 Use Cases, World Markets: 2025*



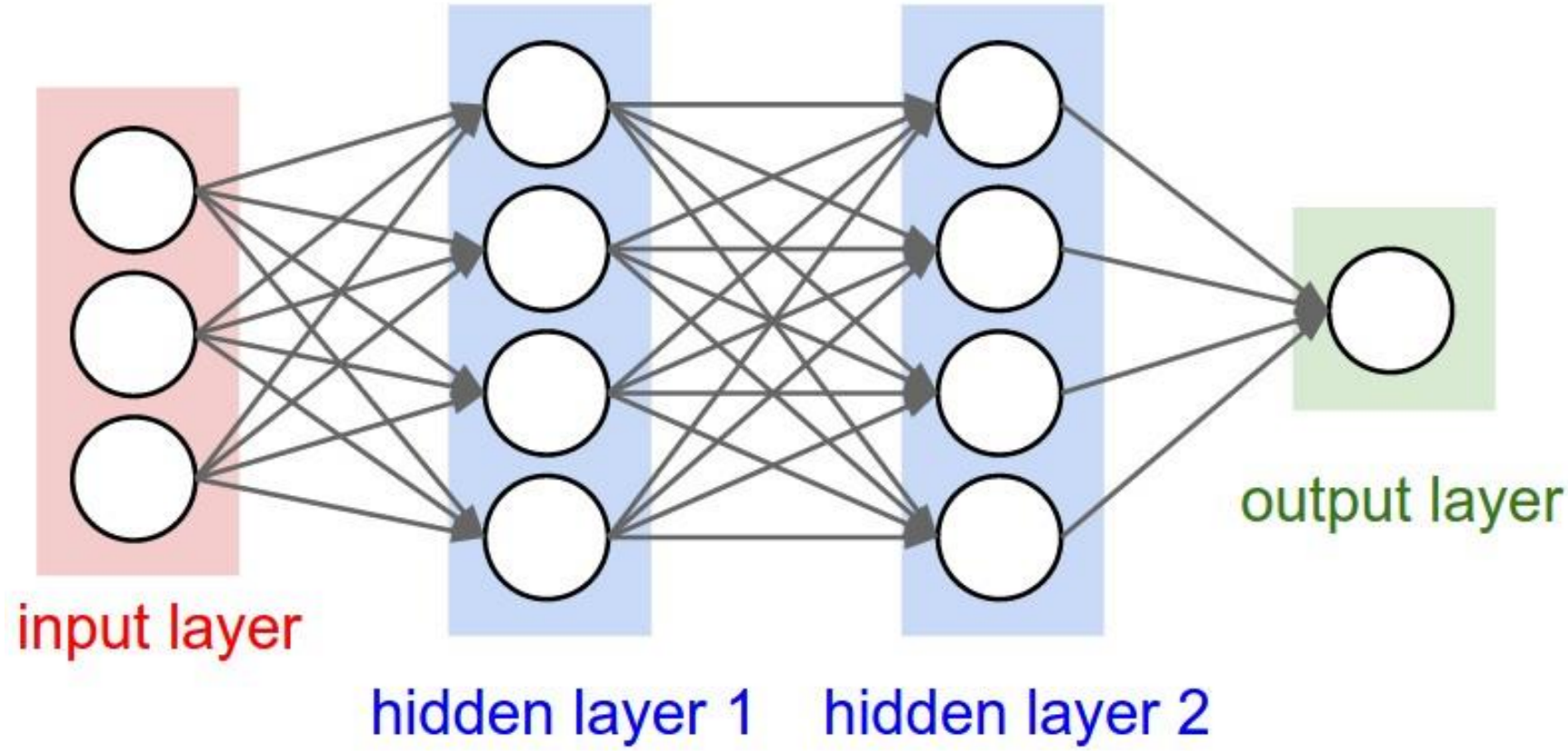
Courtesy: https://www.top500.org/news/market-for-artificial-intelligence-projected-to-hit-36-billion-by-2025/

Outline

- **Introduction**
  - The Past, Present, and Future of Deep Learning
  - **What are Deep Neural Networks?**
  - Diverse Applications of Deep Learning
  - Deep Learning Frameworks
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Conclusion

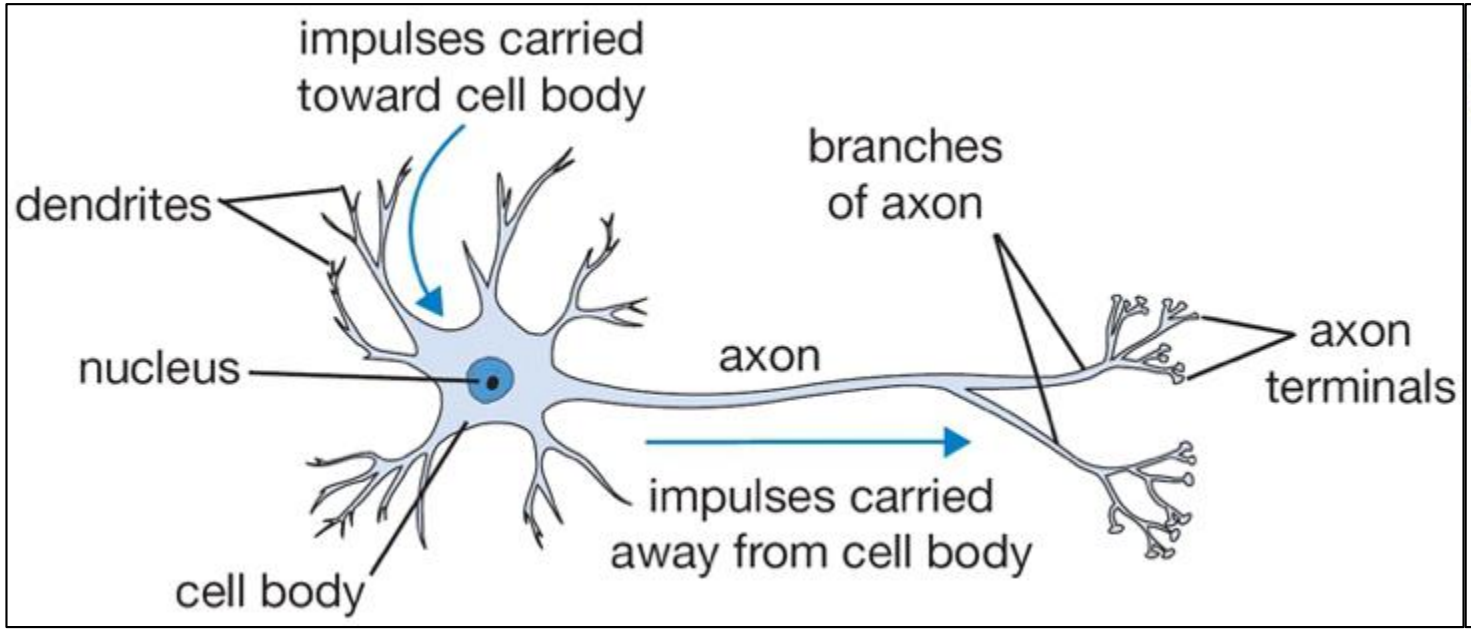
So what is a Deep Neural Network?

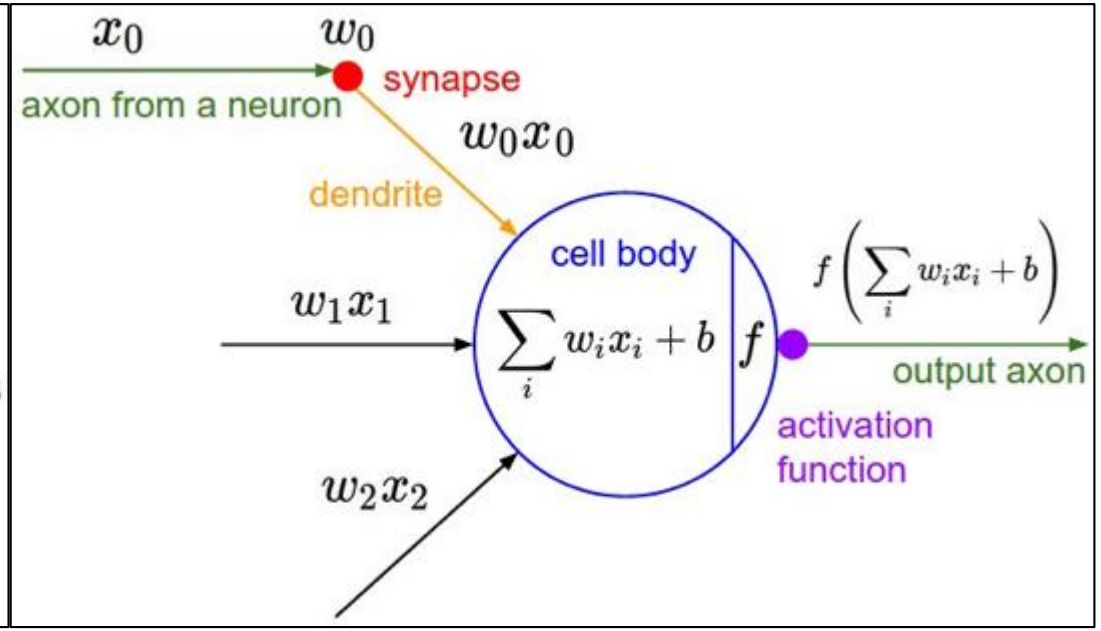
- Example of a 3-layer Deep Neural Network (DNN) – (input layer is not counted)



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

Graphical/Mathematical Intuitions for DNNs





Drawing of a Biological Neuron

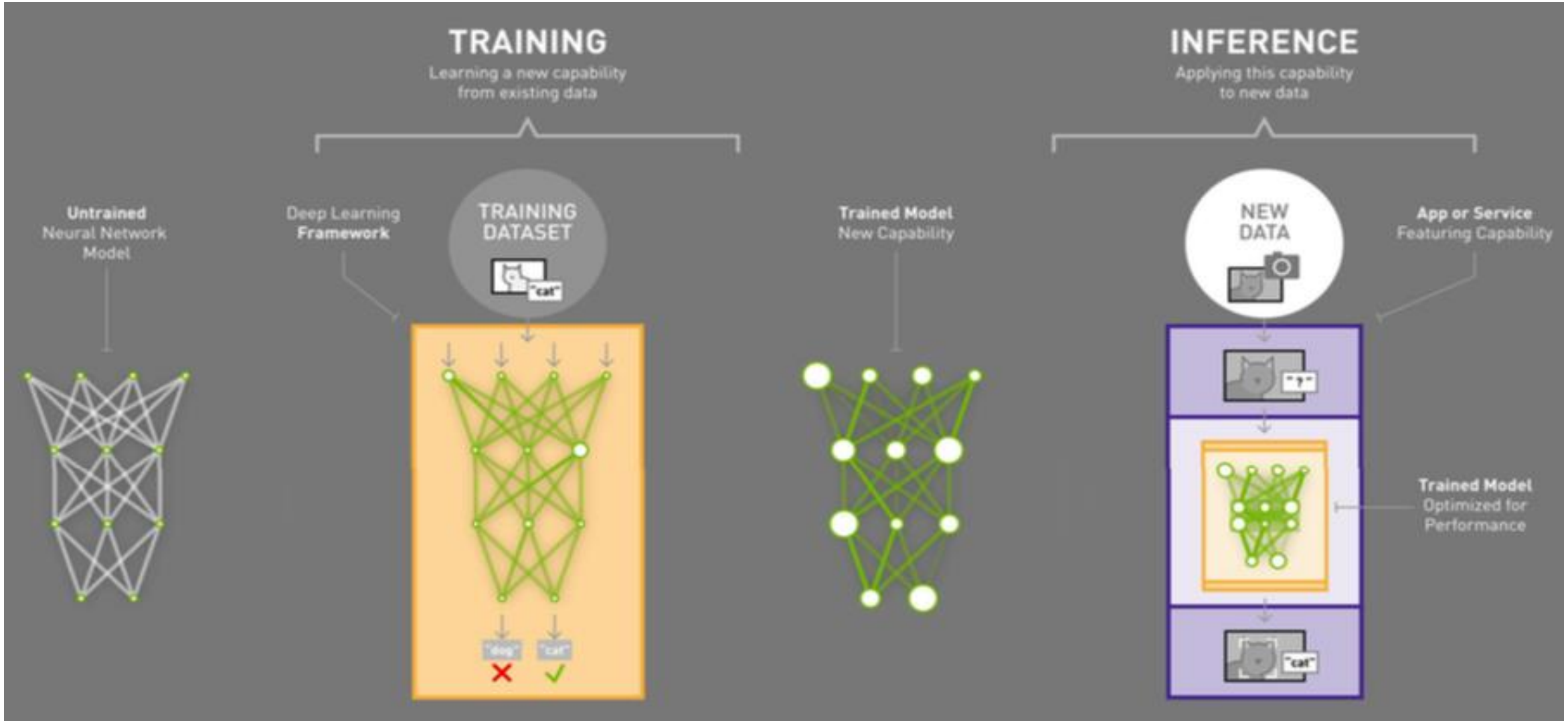
The Mathematical Model

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

Key Phases of Deep Learning

- Deep Learning has two major tasks
  1. Training of the Deep Neural Network
  2. Inference (or deployment) that uses a trained DNN
- DNN Training
  - Training is a compute/communication intensive process – can take days to weeks
  - Faster training is necessary!
- Faster training can be achieved by
  - Using Newer and Faster Hardware – But, there is a limit!
  - Can we use more GPUs or nodes?
    - The need for Parallel and Distributed Training

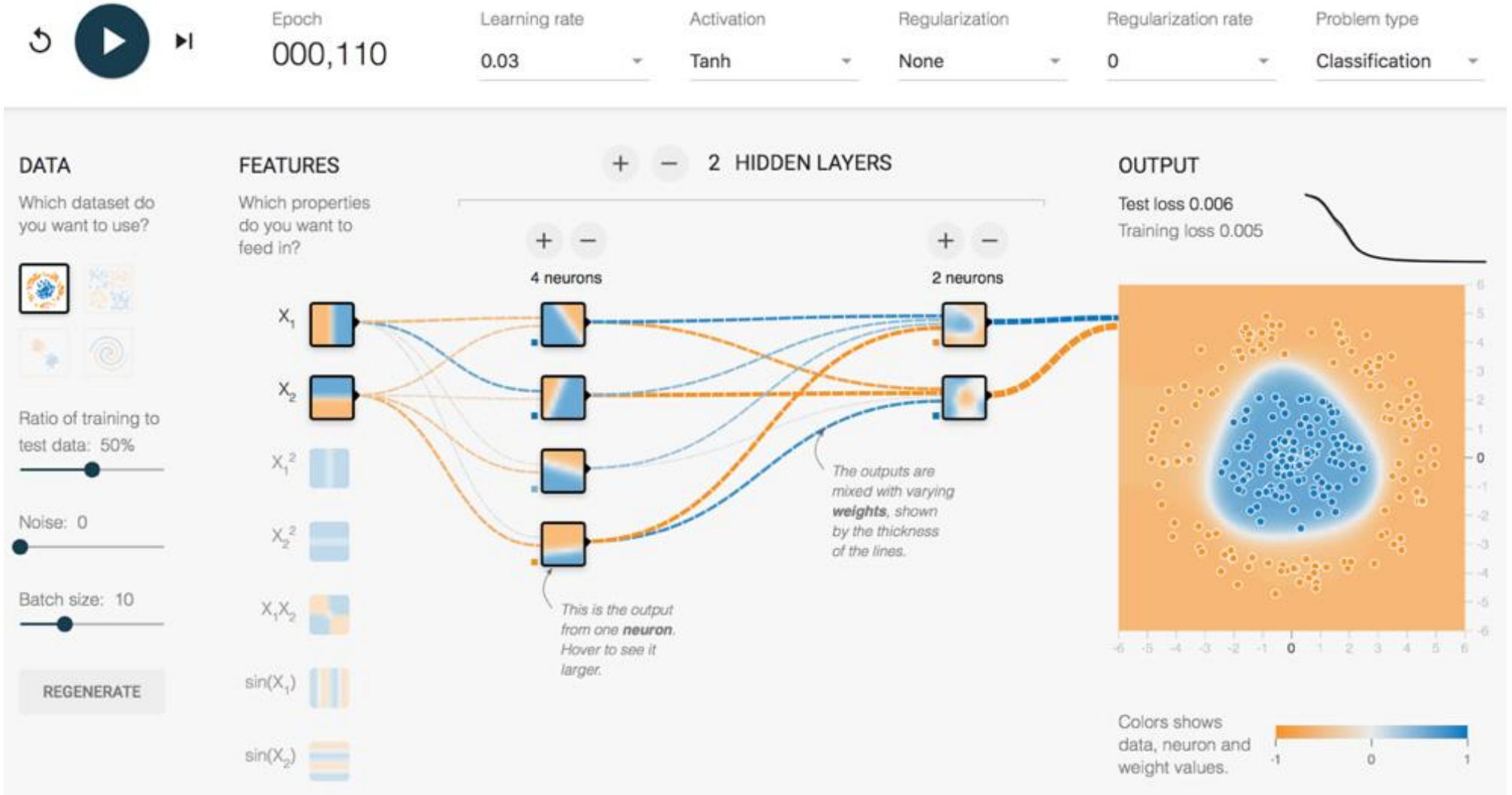
DNN Training and Inference



Courtesy: http://on-demand.gputechconf.com/gtc/2017/presentation/s7457-william-ramey-deep%20learning%20demystified_v24.pdf

TensorFlow playground (Quick Demo)

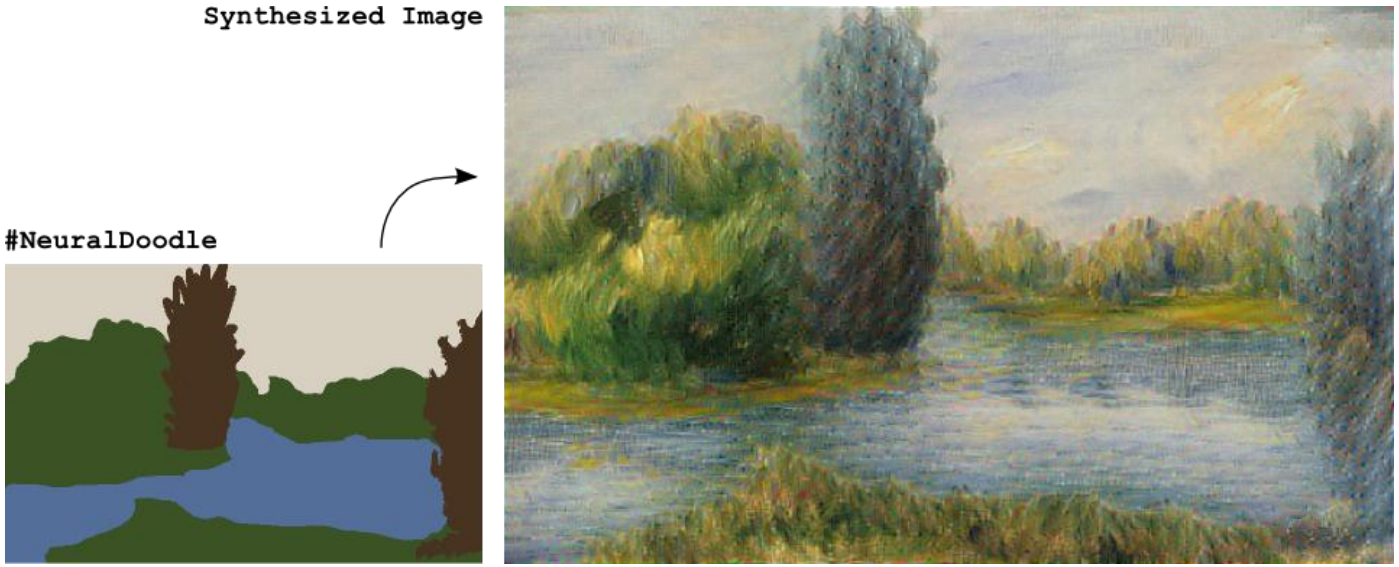
- To actually train a network, please visit: http://playground.tensorflow.org



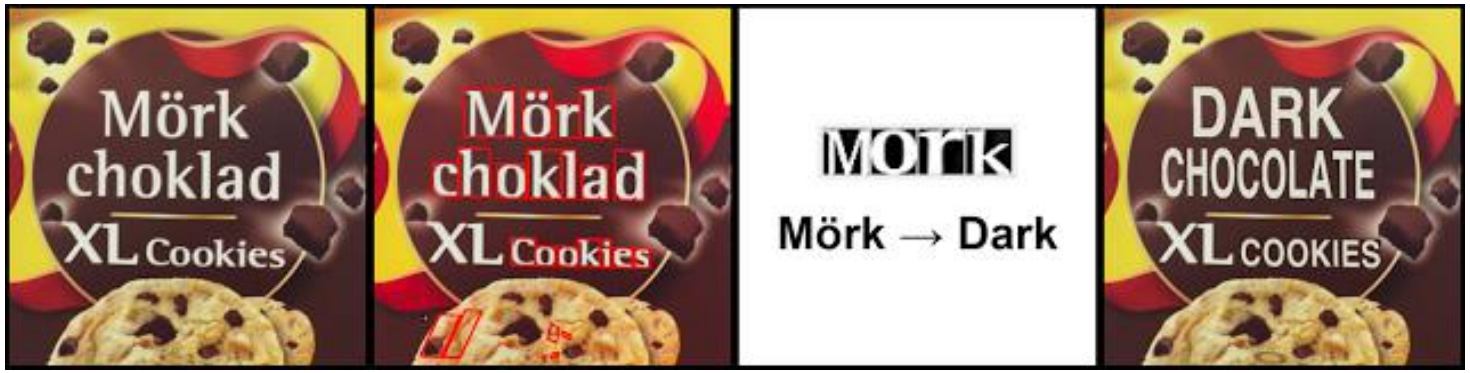
Outline

- **Introduction**
  - The Past, Present, and Future of Deep Learning
  - What are Deep Neural Networks?
  - **Diverse Applications of Deep Learning**
  - Deep Learning Frameworks
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Conclusion

Caption Generation, Translation, Style Transfer, and many more..



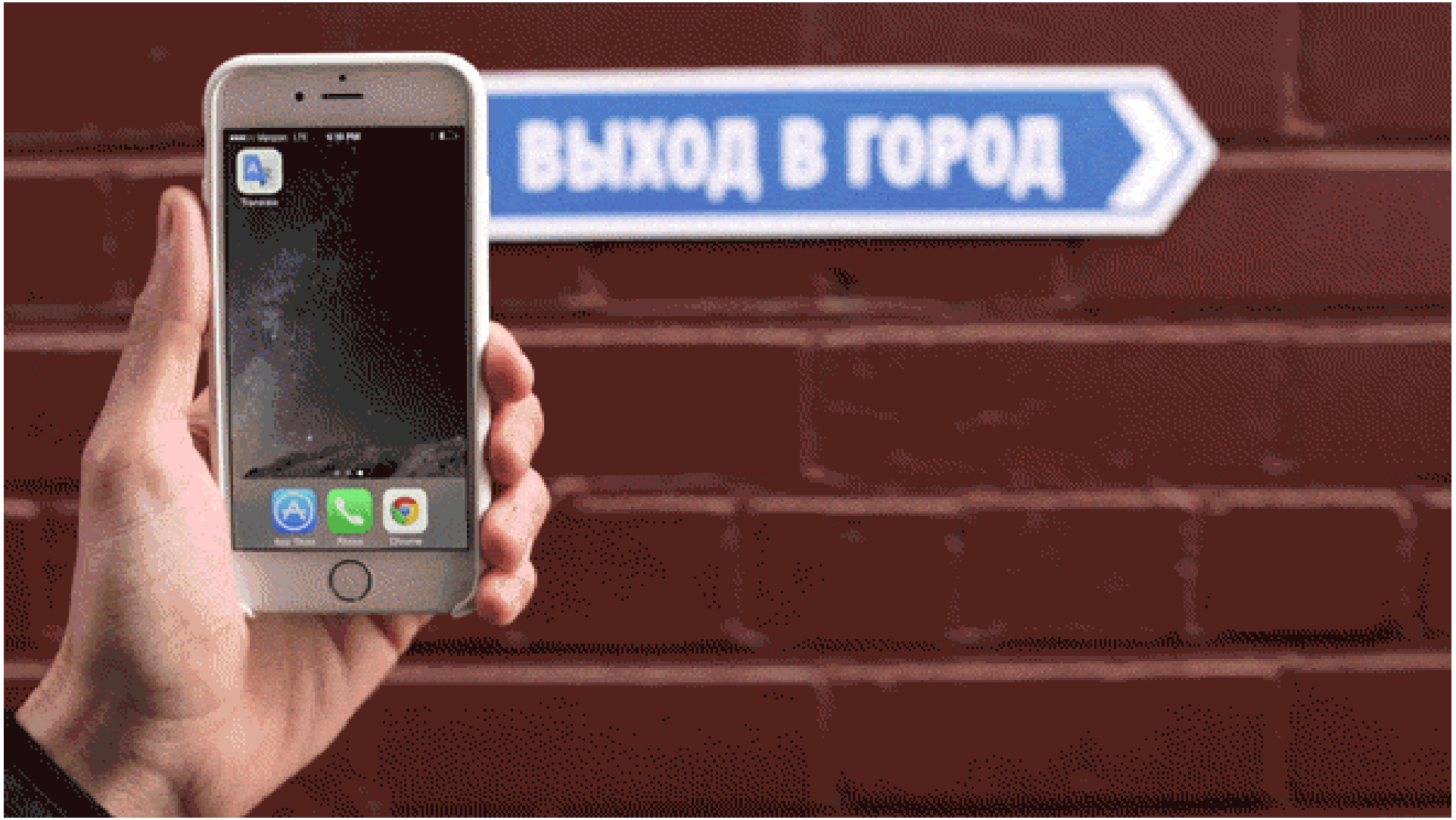
Courtesy: https://github.com/alexjc/neural-doodle



Courtesy: https://research.googleblog.com/2015/07/how-google-translate-squeezes-deep.html

Courtesy: https://machinelearningmastery.com/inspirational-applications-deep-learning/

Google Translate



Courtesy: https://www.theverge.com/2015/1/14/7544919/google-translate-update-real-time-signs-conversations

Self Driving Cars

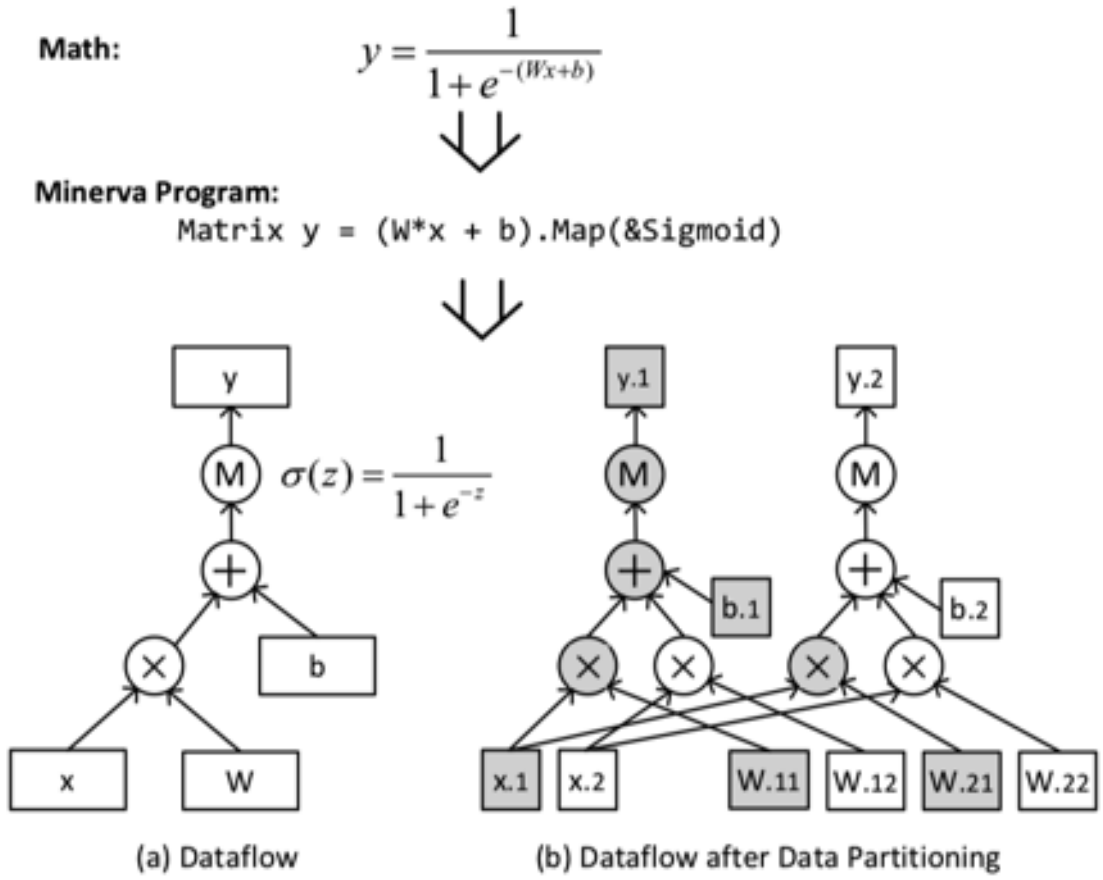
Courtesy: http://www.teslarati.com/teslas-full-self-driving-capability-arrive-3-months-definitely-6-months-says-musk/

Outline

- **Introduction**
  - The Past, Present, and Future of Deep Learning
  - What are Deep Neural Networks?
  - Diverse Applications of Deep Learning
  - **Deep Learning Frameworks**
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Conclusion

Why we need DL frameworks?

- Deep Learning frameworks have emerged
  - hide most of the *nasty mathematics*
  - focus on the *design* of neural networks
- Distributed DL frameworks are being designed
  - We have saturated the peak potential of a single GPU/CPU/KNL
  - Parallel (multiple processing units in a single node) and/or Distributed (usually involves multiple nodes) frameworks are emerging
- Distributed frameworks are being developed along two directions
  - The HPC Eco-system: MPI-based Deep Learning
  - Enterprise Eco-system: BigData-based Deep Learning

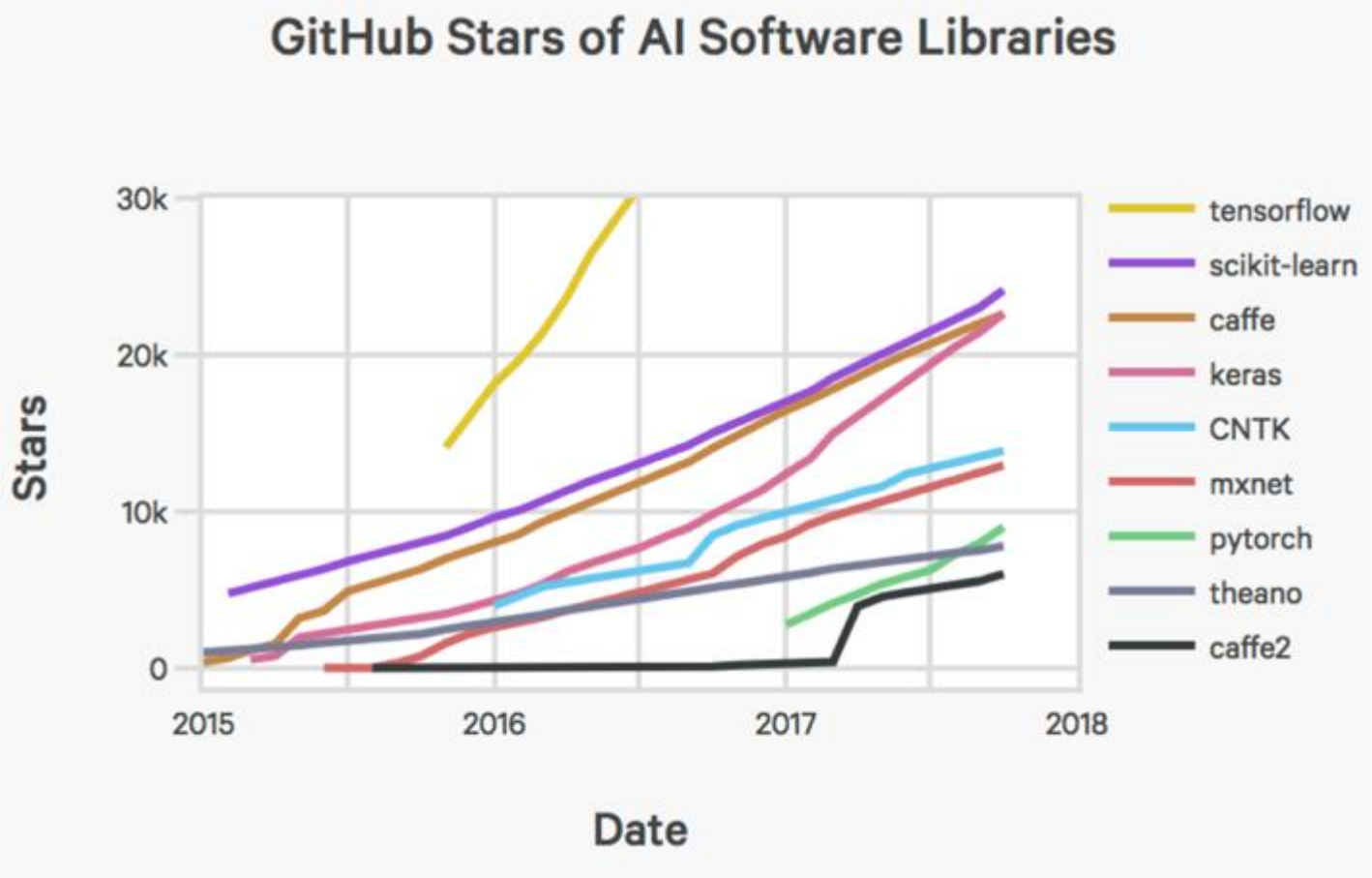


Statement and its dataflow fragment. The data and computing vertexes with different colors reside on different processes.

Courtesy: https://web.stanford.edu/~rezab/nips2014workshop/submits/minerva.pdf

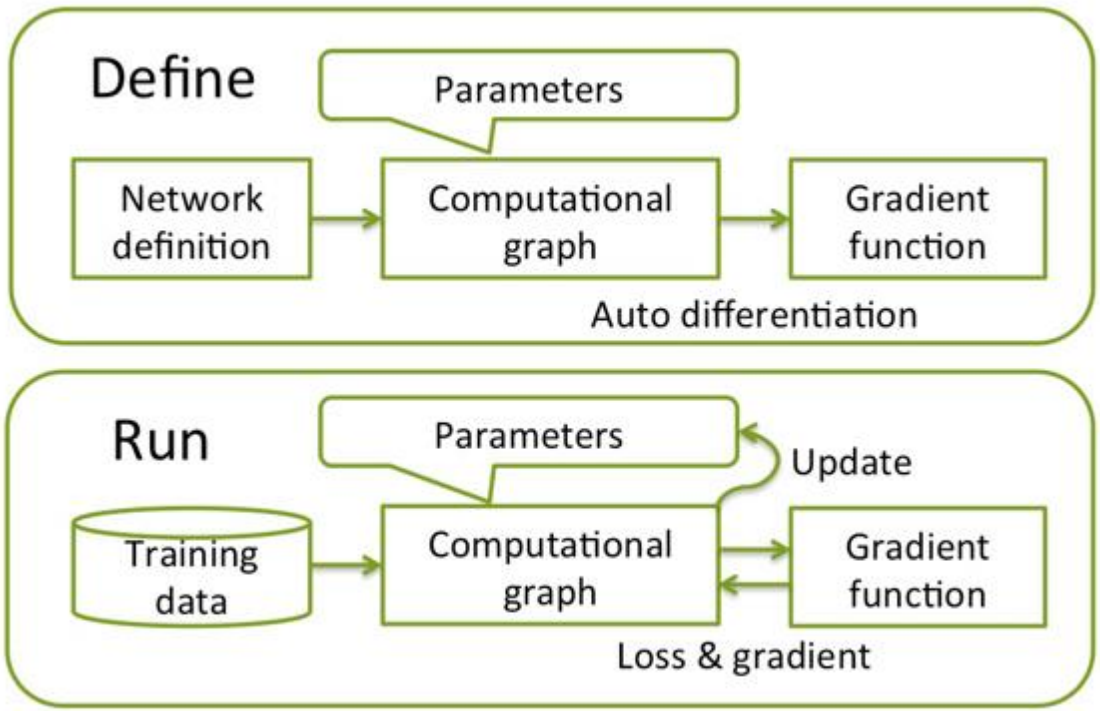
DL Frameworks and GitHub Statistics

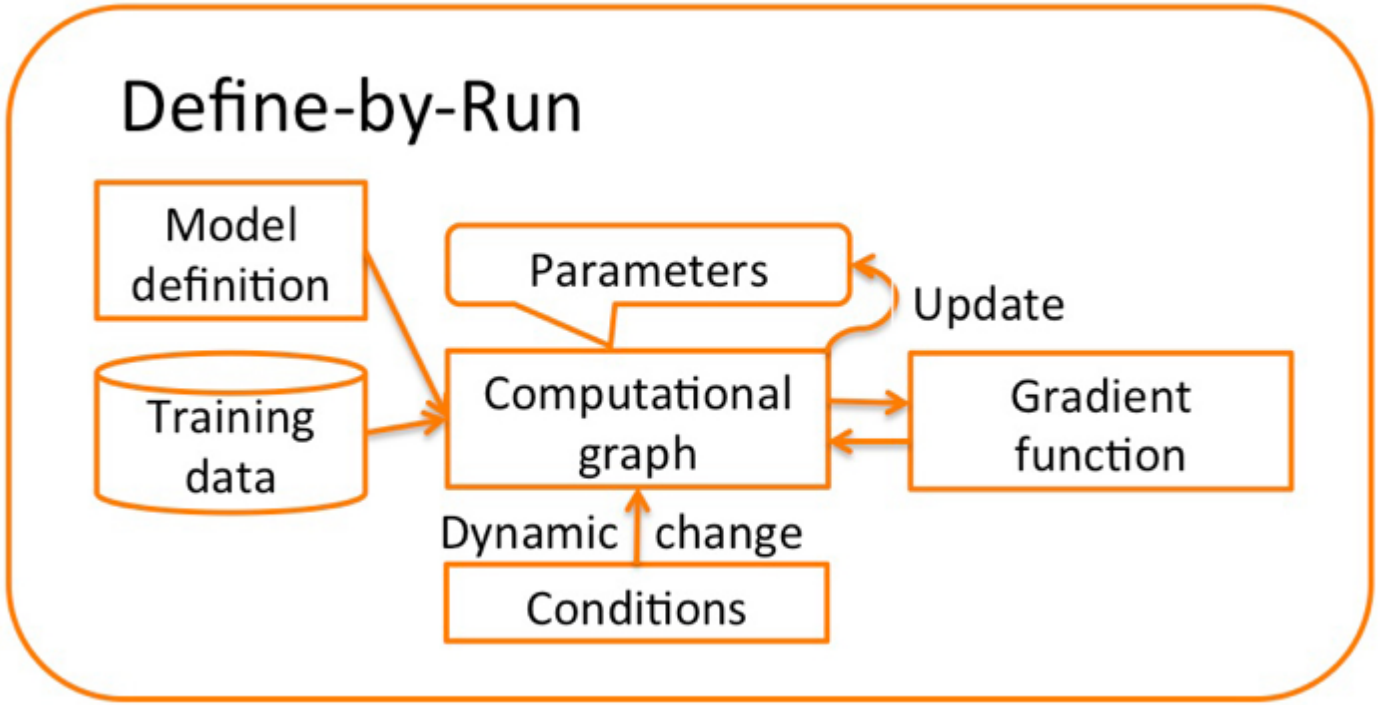
- AI Index report offers very detailed trends about AI and ML
- It also provides interesting statistics about open source DL frameworks and related GitHub statistics



Courtesy: http://cdn.aiindex.org/2017-report.pdf

Are Define-by-run frameworks easier than Define-and-run?





- Define-and-run: TensorFlow, Caffe, Torch, Theano, and others
- Define-by-run
  - PyTorch and Chainer
  - TensorFlow 1.5 introduced Eager Execution (Define-by-run) mode

Courtesy: https://www.oreilly.com/learning/complex-neural-networks-made-easy-by-chainer

Google TensorFlow (Most Popular)

- The most widely used framework open-sourced by Google
- Replaced Google's DistBelief[1] framework
- Runs on almost all execution platforms available (CPU, GPU, TPU, Mobile, etc.)
- Very flexible but performance has been an issue
- Certain Python peculiarities like ***variable_scope*** etc.
- https://github.com/tensorflow/tensorflow

Courtesy: https://www.tensorflow.org/

[1] Jeffrey Dean et al., "Large Scale Distributed Deep Networks"
https://static.googleusercontent.com/media/research.google.com/en//archive/large_deep_networks_nips2012.pdf

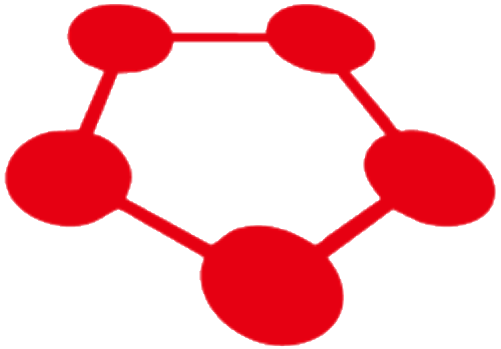
Facebook Torch/PyTorch - Catching up fast!

- Torch was written in Lua
  - Adoption wasn't wide-spread
- PyTorch is a Python adaptation of Torch
  - Gaining lot of attention
- Several contributors
  - Biggest support by Facebook
- There are/maybe plans to merge the PyTorch and Caffe2 efforts
- Key selling point is ease of expression and "define-by-run" approach

Courtesy: http://pytorch.org

Preferred Networks Chainer/ChainerMN

- ChainerMN provides multi-node parallel/distributed training using MPI
  - **MVAPICH2** MPI library is being used by Preferred Networks
  - **http://mvapich.cse.ohio-state.edu**



- ChainerMN is geared towards performance
  - Uses ***Define-by-run*** (Chainer, PyTorch) approach instead of ***Define-and-run*** (Caffe, TensorFlow, Torch, Theano) approach
  - https://github.com/chainer/chainer
  - Focus on Speed as well as multi-node Scaling
  - Beats CNTK, MXNet, and TensorFlow for training ResNet-50 on 128 GPUs [1]

1. http://chainer.org/general/2017/02/08/Performance-of-Distributed-Deep-Learning-Using-ChainerMN.html

Many Other DL Frameworks…

- Keras - https://keras.io
- MXNet - http://mxnet.io
- Theano - http://deeplearning.net/software/theano/
- Blocks - https://blocks.readthedocs.io/en/latest/
- Intel BigDL - https://software.intel.com/en-us/articles/bigdl-distributed-deep-learning-on-apache-spark
- The list keeps growing and the names keep getting longer and weirder ;-)
  - Livermore Big Artificial Neural Network Toolkit (LBANN) - https://github.com/LLNL/lbann
  - Deep Scalable Sparse Tensor Network Engine (DSSTNE) - https://github.com/amzn/amazon-dsstne

Outline

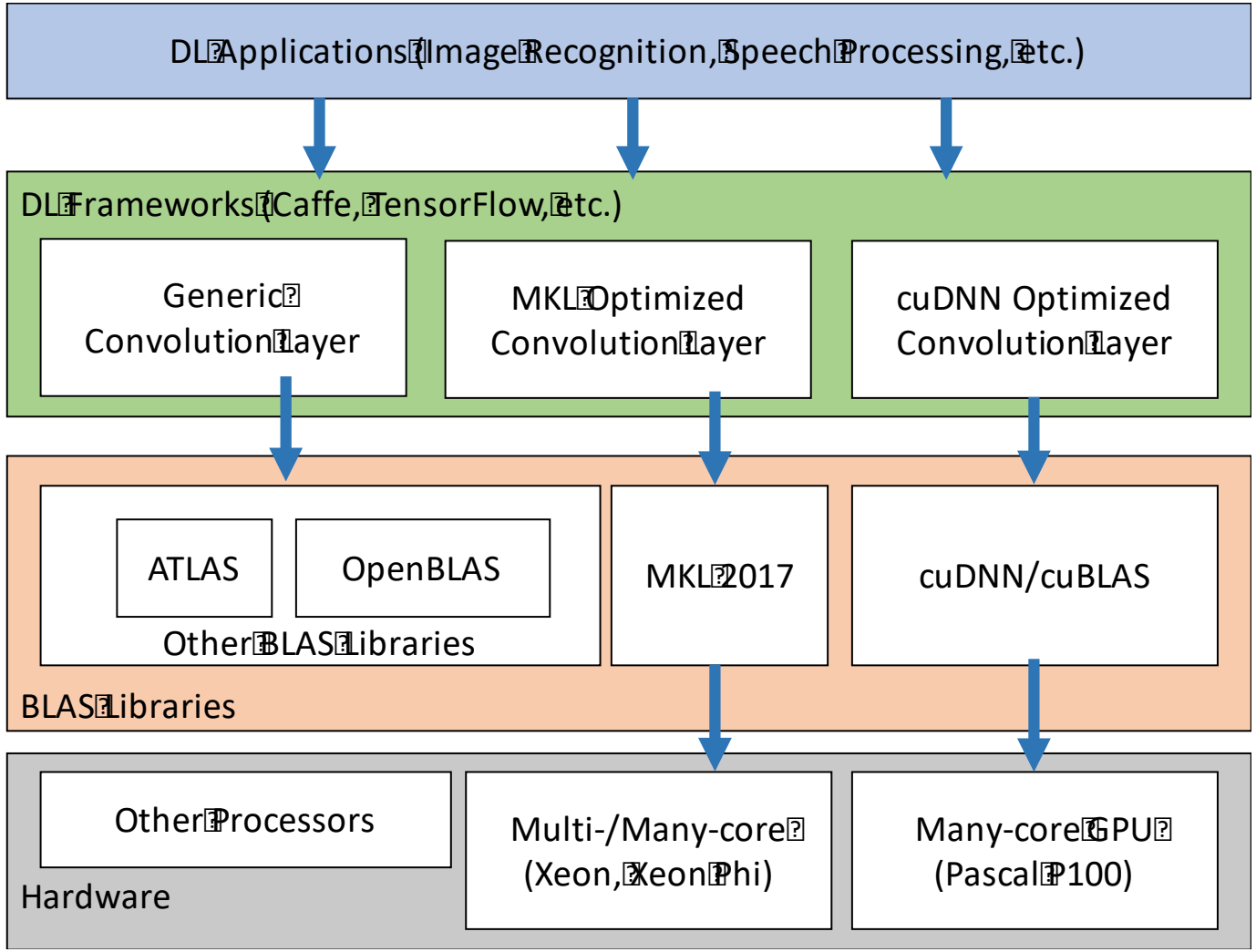
- Introduction
- **Overview of Execution Environments**
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Conclusion

So where do we run our DL framework?

- Early (2014) frameworks used a single fast GPU
  - As DNNs became larger, faster and better GPUs became available
  - At the same time, parallel (multi-GPU) training gained traction as well
- Today
  - Parallel training on multiple GPUs is being supported by most frameworks
  - Distributed (multiple nodes) training is still upcoming
    - A lot of fragmentation in the efforts (MPI, Big-Data, NCCL, Gloo, etc.)
  - On the other hand, DL has made its way to Mobile and Web too!
    - Smartphones - OK Google, Siri, Cortana, Alexa, etc.
    - DrivePX – the computer that drives NVIDIA's self-driving car
    - Deeplearn.js – a DL framework in a web-browser
    - TensorFlow playground - http://playground.tensorflow.org/

Conventional Execution on GPUs and CPUs

- My framework is faster than your framework!
- This needs to be understood in a holistic way.
- Performance depends on the entire execution environment (the full stack)
- Isolated view of performance is not helpful

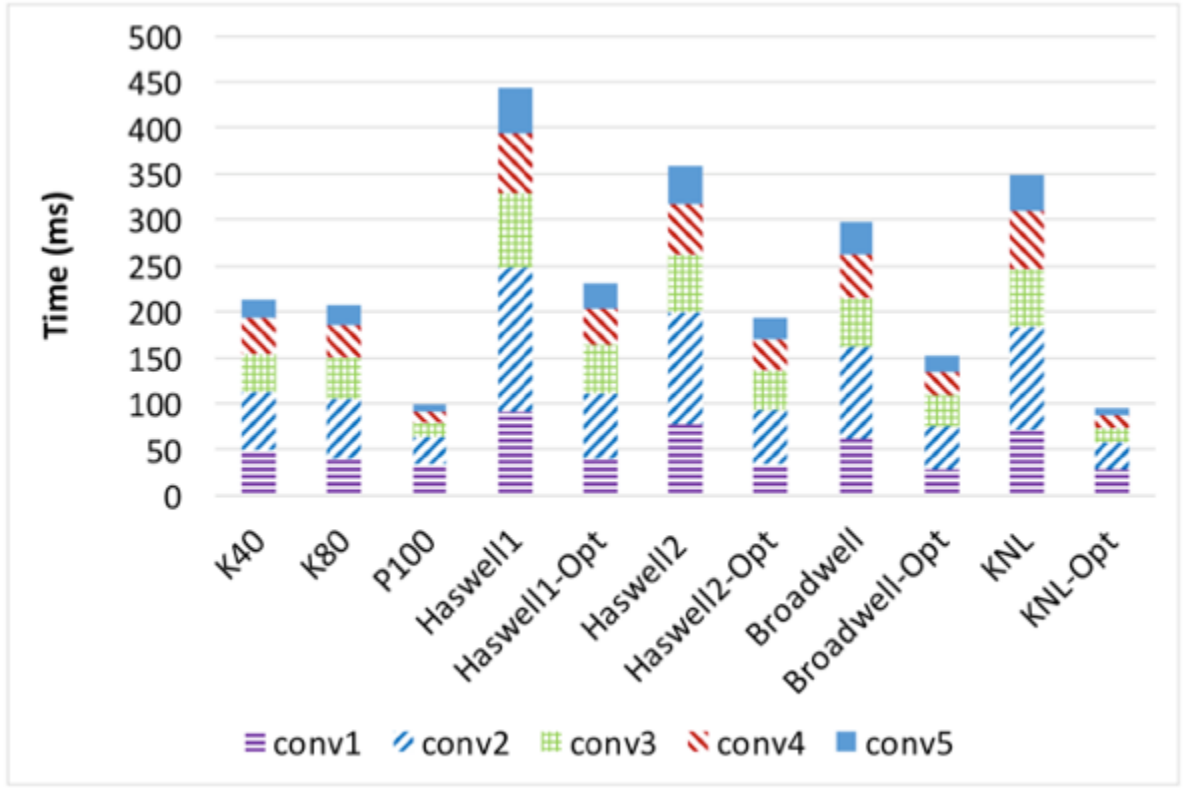


A. A. Awan, H. Subramoni, and Dhabaleswar K. Panda. "An In-depth Performance Characterization of CPU- and GPU-based DNN Training on Modern Architectures", In Proceedings of the Machine Learning on HPC Environments (MLHPC'17). ACM, New York, NY, USA, Article 8.

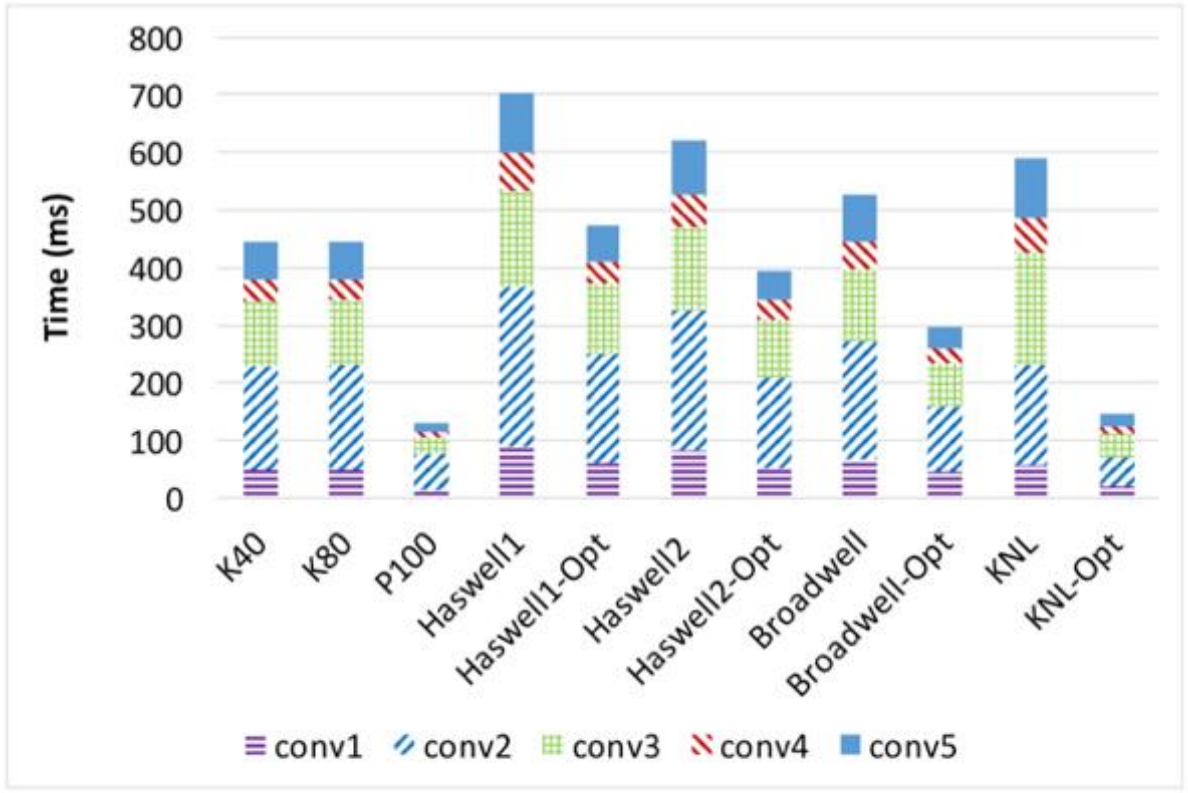
DL Frameworks and Underlying Libraries

- BLAS Libraries – the heart of math operations
  - Atlas/OpenBLAS
  - NVIDIA cuBlas
  - Intel Math Kernel Library (MKL)
- Most compute intensive layers are generally optimized for a specific hardware
  - E.g. Convolution Layer, Pooling Layer, etc.
- DNN Libraries – the heart of Convolutions!
  - NVIDIA cuDNN (already reached its 7th iteration – cudnn-v7.5)
  - Intel MKL-DNN (MKL 2018) – recent but a very promising development

Where does the Performance come from?



(a) AlexNet: Forward Propagation



(b) AlexNet: Backward Propagation

- The full landscape: Forward and Backward Pass -- ***Faster Convolutions → Faster Training***
- Performance of Intel KNL == NVIDIA P100 for AlexNet Training – ***Volta is in a different league!***
- Most performance gains are based on improvements in layer ***conv2*** and ***conv3*** for AlexNet

A. A. Awan, H. Subramoni, and Dhabaleswar K. Panda. "An In-depth Performance Characterization of CPU- and GPU-based DNN Training on Modern Architectures", In Proceedings of the Machine Learning on HPC Environments (MLHPC'17). ACM, New York, NY, USA, Article 8.

Outline

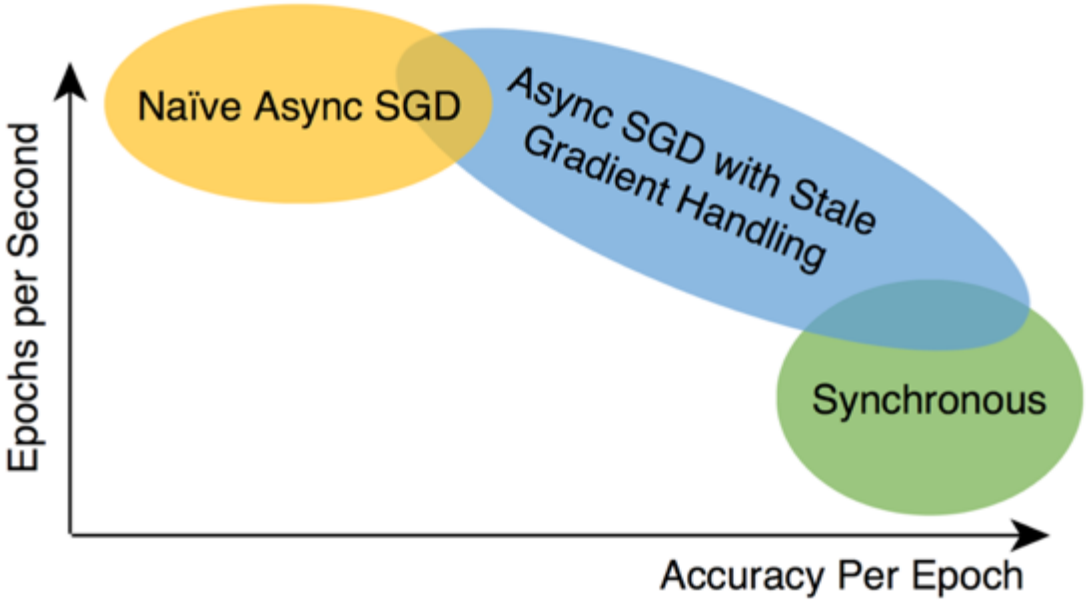
- Introduction
- Overview of Execution Environments
- **Parallel and Distributed DNN Training**
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Conclusion

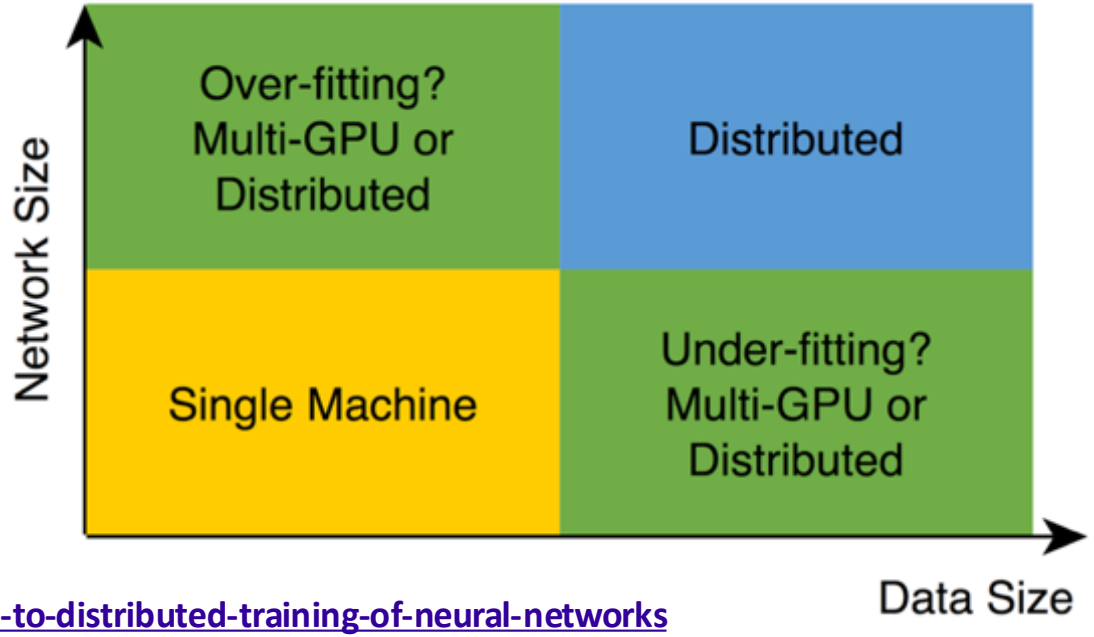
The Need for Parallel and Distributed Training

- Why do we need Parallel Training?
- Larger and Deeper models are being proposed
  - **AlexNet** to **ResNet** to **Neural Machine Translation (NMT)**
  - DNNs require a lot of memory
  - Larger models cannot fit a GPU's memory
- Single GPU training became a bottleneck
- As mentioned earlier, community has already moved to multi-GPU training
- Multi-GPU in one node is good but there is a limit to Scale-up (8 GPUs)
- **Multi-node (Distributed or Parallel) Training is necessary!!**

Batch-size, Model-size, Accuracy, and Scalability

- Increasing model-size generally increases accuracy
- Increasing batch-size requires tweaking hyper-parameters to maintain accuracy
  - Limits for batch-size
  - Cannot make it infinitely large
  - Over-fitting
- **Large batch size generally helps scalability**
  - More work to do before the need to synchronize
- Increasing the model-size (no. of parameters)
  - Communication overhead becomes bigger so scalability decreases
  - GPU memory is precious and can only fit finite model data





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

Benefits of Distributed Training: An Example with Caffe

- Strong scaling CIFAR10 Training with OSU-Caffe (1 –> 4 GPUs) – **Batch Size 2K**
- Large batch size is needed for scalability.
- Adding more GPUs will degrade the scaling efficient

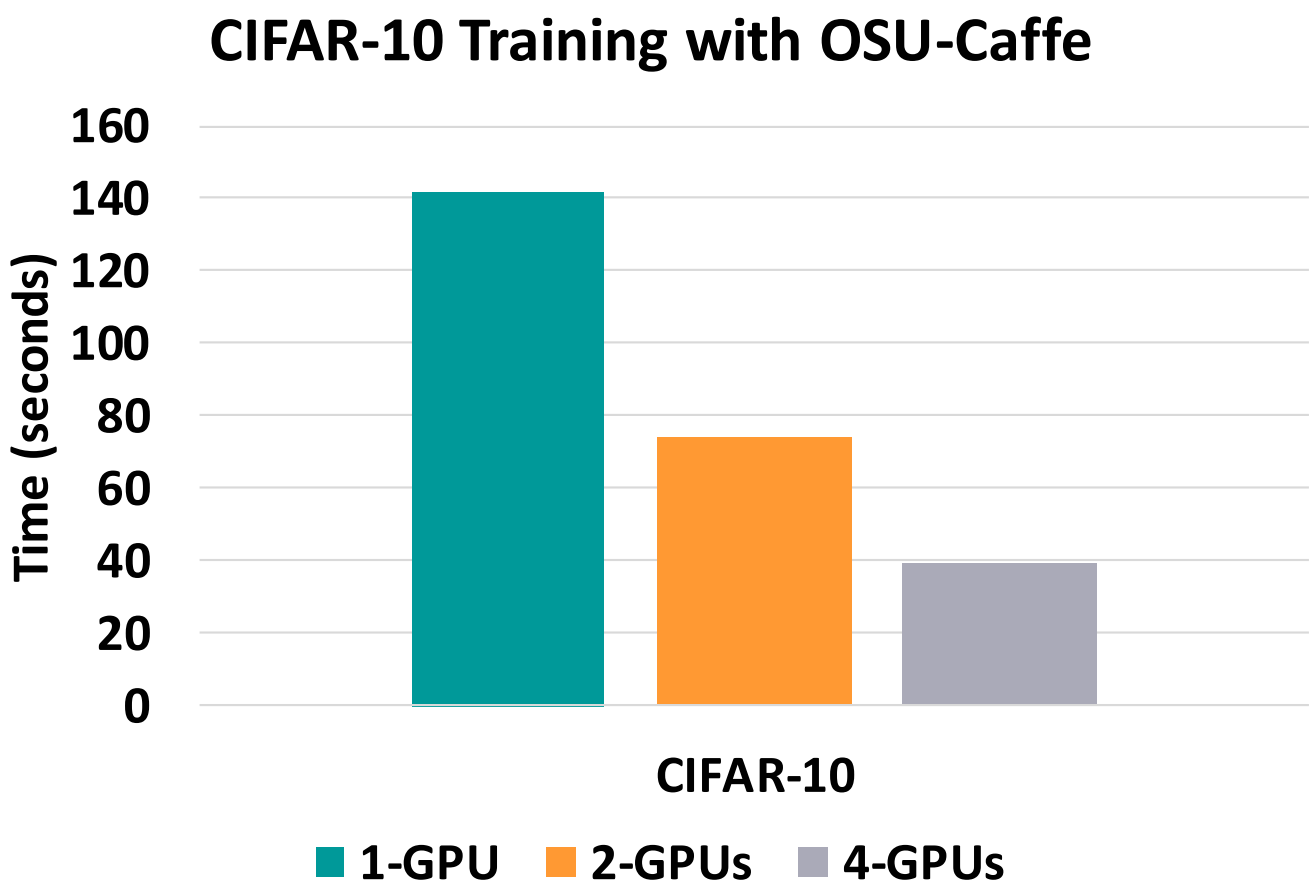
Run Command - (change $np from 1—4)

mpirun_rsh -np **$np** ./build/tools/caffe train -solver examples/cifar10/cifar10_quick_solver.prototxt -scal **strong**

Output: I0123 21:49:24.289763 75582 caffe.cpp:351] Avg. Time Taken: 142.101

Output: I0123 21:54:03.449211 97694 caffe.cpp:351] Avg. Time Taken: 74.6679

Output: I0123 22:02:46.858219 20659 caffe.cpp:351] Avg. Time Taken: 39.8109

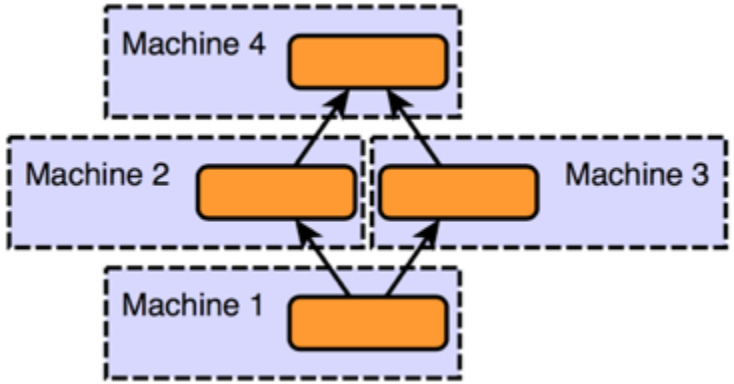


OSU-Caffe is available from the HiDL project page (http://hidl.cse.ohio-state.edu)

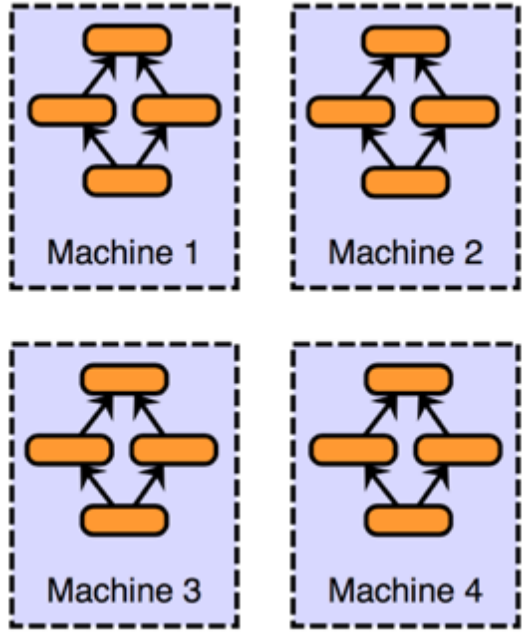
Parallelization Strategies

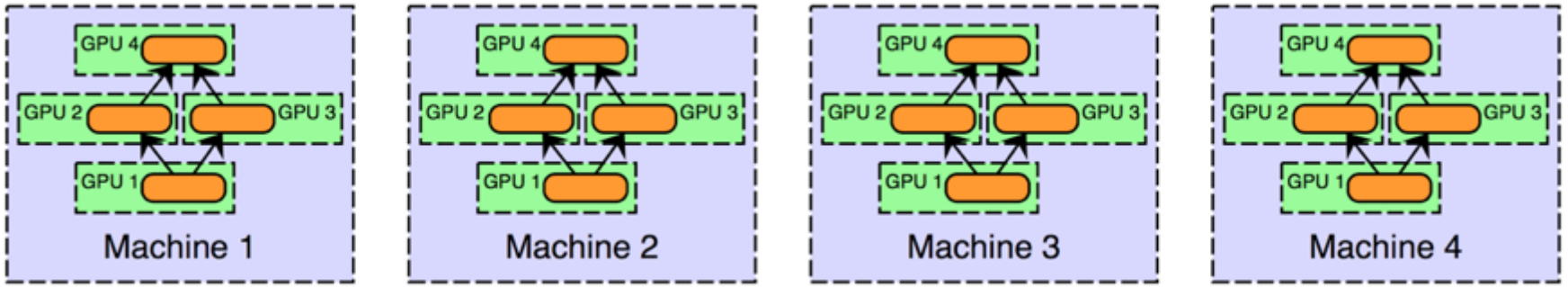
- What are the Parallelization Strategies
  - Model Parallelism
  - **Data Parallelism (Received the most attention)**
  - Hybrid Parallelism
  - Automatic Selection

Model Parallelism



Data Parallelism





Hybrid (Model and Data) Parallelism

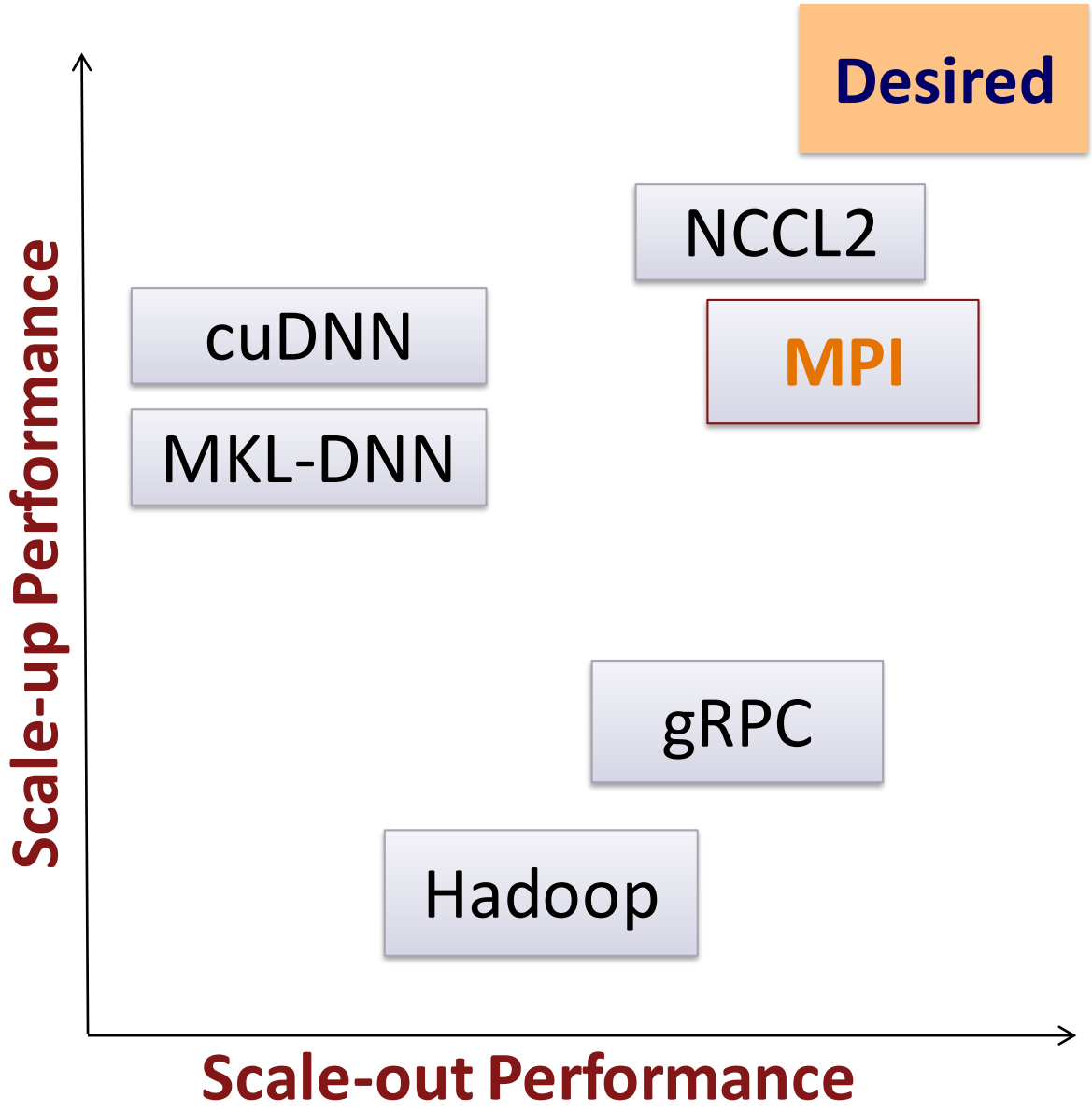
Courtesy: http://engineering.skymind.io/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks

Communication in Distributed Frameworks

- What are the Design Choices for Communication?
  - Established paradigms like Message Passing Interface (MPI)
  - Develop specific communication libraries like NCCL, Gloo, Baidu-allreduce, etc.
  - Use Big-Data frameworks like Spark, Hadoop, etc.
    - Still need some form of external communication for parameters (RDMA, IB, etc.)
- Focus on Scale-up and Scale-out
  - What are the challenges and opportunities?

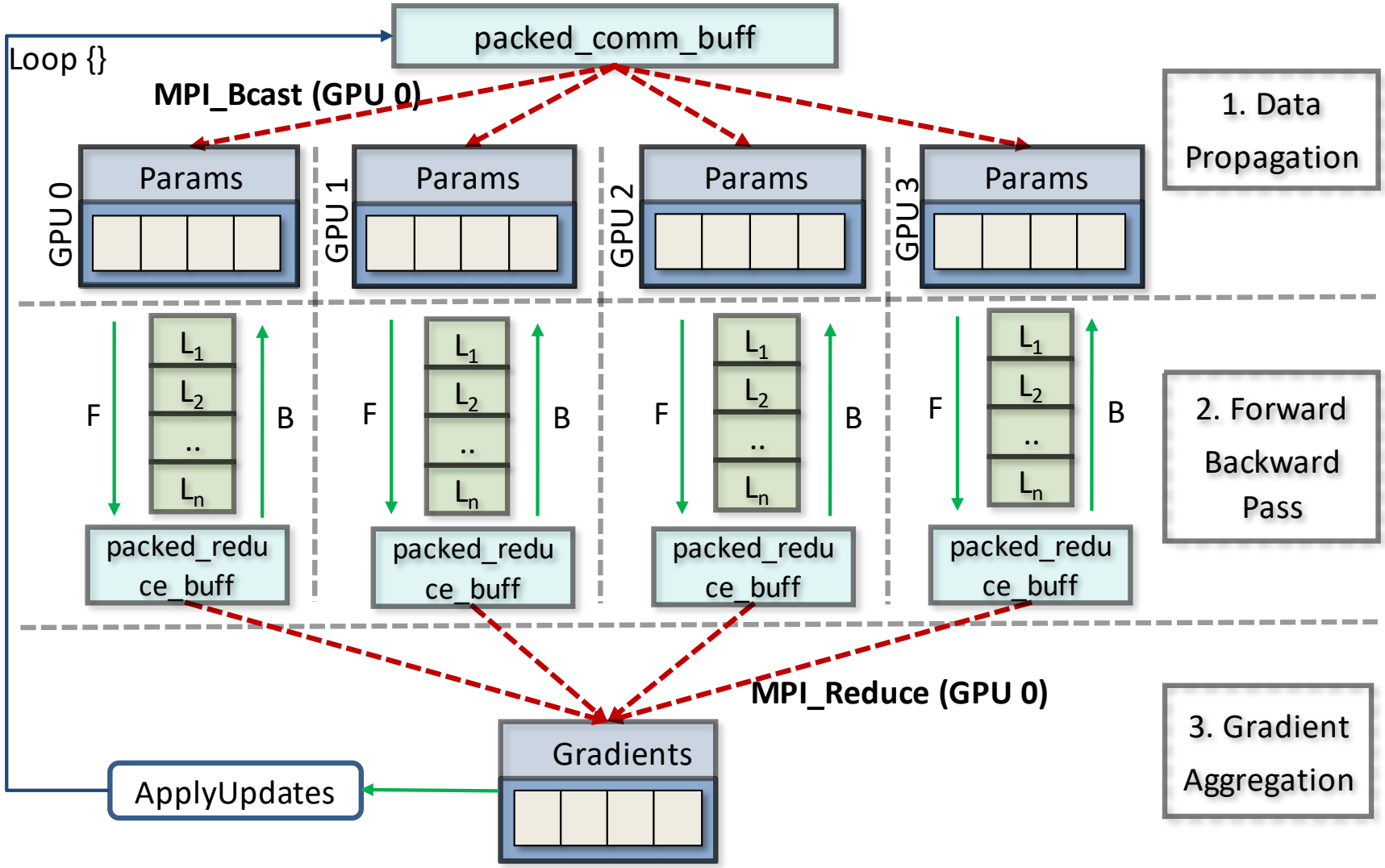
Scale-up and Scale-out

- **Scale-up**: Intra-node Communication
  - Many improvements like:
    - NVIDIA cuDNN, cuBLAS, NCCL, etc.
    - CUDA 9 Co-operative Groups
- **Scale-out**: Inter-node Communication
  - DL Frameworks – most are optimized for single-node only
  - Distributed (Parallel) Training is an emerging trend
    - **OSU-Caffe – MPI-based**
    - Microsoft CNTK – MPI/NCCL2
    - Google TensorFlow – gRPC-based/MPI/NCCL2
    - Facebook Caffe2 – Hybrid (NCCL2/Gloo/MPI)



Data Parallel Deep Learning and MPI Collectives

- Major **MPI Collectives** involved in Designing distributed frameworks
- **MPI_Bcast** – required for DNN parameter exchange
- **MPI_Reduce** – needed for gradient accumulation from multiple solvers
- **MPI_Allreduce** – use just one Allreduce instead of Reduce and Broadcast



A. A. Awan, K. Hamidouche, J. M. Hashmi, and D. K. Panda, S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters. In *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (PPoPP '17)

Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- **Latest Trends in HPC Technologies**
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Conclusion

Drivers of Modern HPC Cluster Architectures

Multi-/Many-core Processors

High Performance Interconnects - InfiniBand
<1usec latency, 100Gbps Bandwidth>

Accelerators
high compute density, high performance/watt
>1 TFlop DP on a chip

SSD, NVMe-SSD, NVRAM

- Multi-core/many-core technologies
- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD
- Accelerators (NVIDIA GPGPUs)
- Available on HPC Clouds, e.g., Amazon EC2, NSF Chameleon, Microsoft Azure, etc.

Summit

Sunway TaihuLight

Sierra

K - Computer

HPC Technologies

- **Hardware**
  - **Interconnects – InfiniBand, RoCE, Omni-Path, etc.**
  - **Processors – GPUs, Multi-/Many-core CPUs, Tensor Processing Unit (TPU), FPGAs, etc.**
- Communication Middleware
  - Message Passing Interface (MPI)
    - CUDA-Aware MPI, Many-core Optimized MPI runtimes (KNL-specific optimizations)
  - NVIDIA NCCL

Overview of High Performance Interconnects

- High-Performance Computing (HPC) has adopted advanced interconnects and protocols
  - InfiniBand (IB)
  - Omni-Path
  - High Speed Ethernet 10/25/40/50/100 Gigabit Ethernet/iWARP
  - RDMA over Converged Enhanced Ethernet (RoCE)
- Very Good Performance
  - Low latency (few micro seconds)
  - High Bandwidth (200 Gb/s with HDR InfiniBand)
  - Low CPU overhead (5-10%)
- OpenFabrics software stack with IB, Omni-Path, iWARP and RoCE interfaces are driving HPC systems
- Many such systems in Top500 list

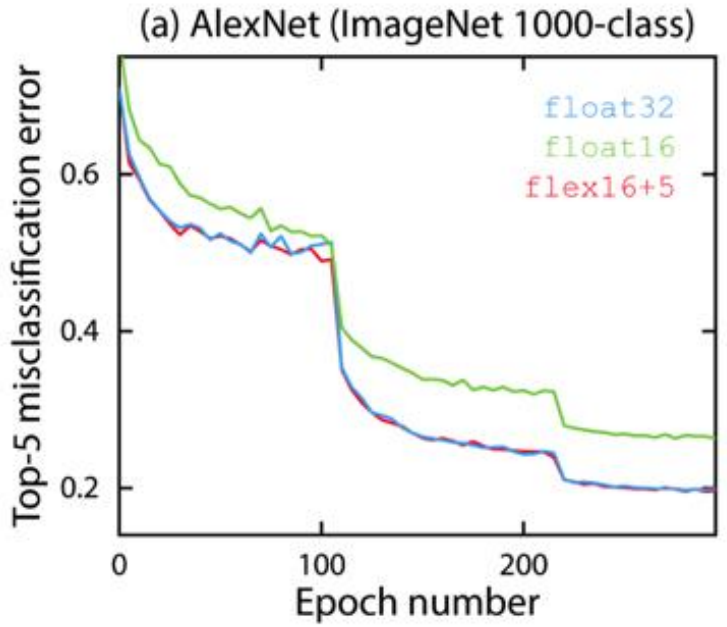
Network Speed Acceleration with IB and HSE

Ethernet (1979 -)	10 Mbit/sec
Fast Ethernet (1993 -)	100 Mbit/sec
Gigabit Ethernet (1995 -)	1000 Mbit /sec
ATM (1995 -)	155/622/1024 Mbit/sec
Myrinet (1993 -)	1 Gbit/sec
Fibre Channel (1994 -)	1 Gbit/sec
InfiniBand (2001 -)	2 Gbit/sec (1X SDR)
10-Gigabit Ethernet (2001 -)	10 Gbit/sec
InfiniBand (2003 -)	8 Gbit/sec (4X SDR)
InfiniBand (2005 -)	16 Gbit/sec (4X DDR)
	24 Gbit/sec (12X SDR)
InfiniBand (2007 -)	32 Gbit/sec (4X QDR)
40-Gigabit Ethernet (2010 -)	40 Gbit/sec
InfiniBand (2011 -)	54.6 Gbit/sec (4X FDR)
InfiniBand (2012 -)	2 x 54.6 Gbit/sec (4X Dual-FDR)
25-/50-Gigabit Ethernet (2014 -)	25/50 Gbit/sec
100-Gigabit Ethernet (2015 -)	100 Gbit/sec
Omni-Path (2015 -)	100 Gbit/sec
InfiniBand (2015 -)	100 Gbit/sec (4X EDR)
InfiniBand (2018 -)	200 Gbit/sec (4X HDR)

100 times in the last 17 years

Intel Neural Network Processor (NNP)

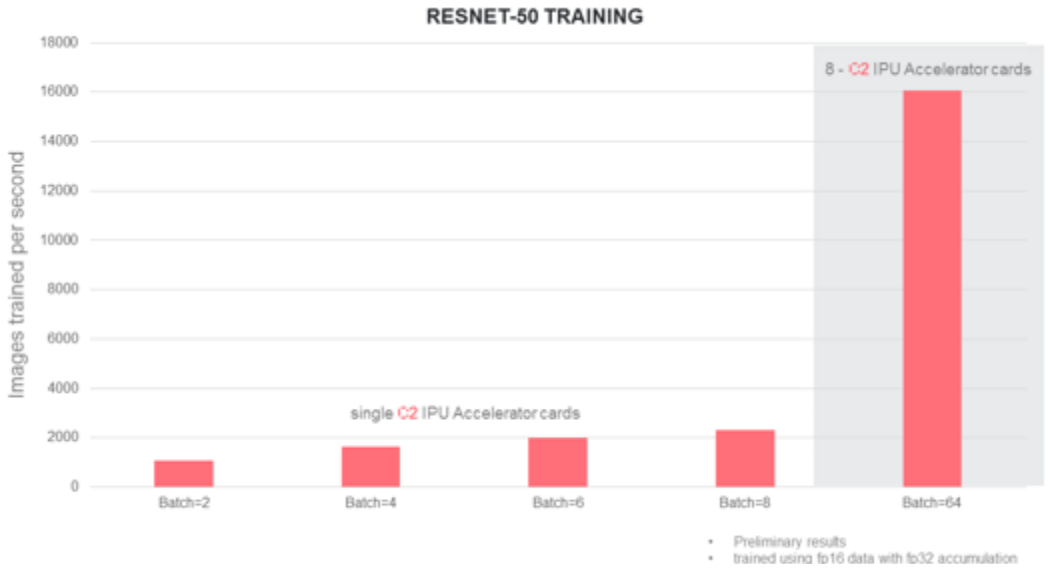
- Intel® Nervana™ Neural Network Processors (NNP)
  - formerly known as “Lake Crest”
- Recently announced as part of Intel’s strategy for next-generation AI systems
- Purpose built architecture for deep learning
- 1 TB/s High Bandwidth Memory (HBM)
- Spatial Architecture
- FlexPoint format
  - Similar performance (in terms of accuracy) to FP32 while using 16 bits of storage

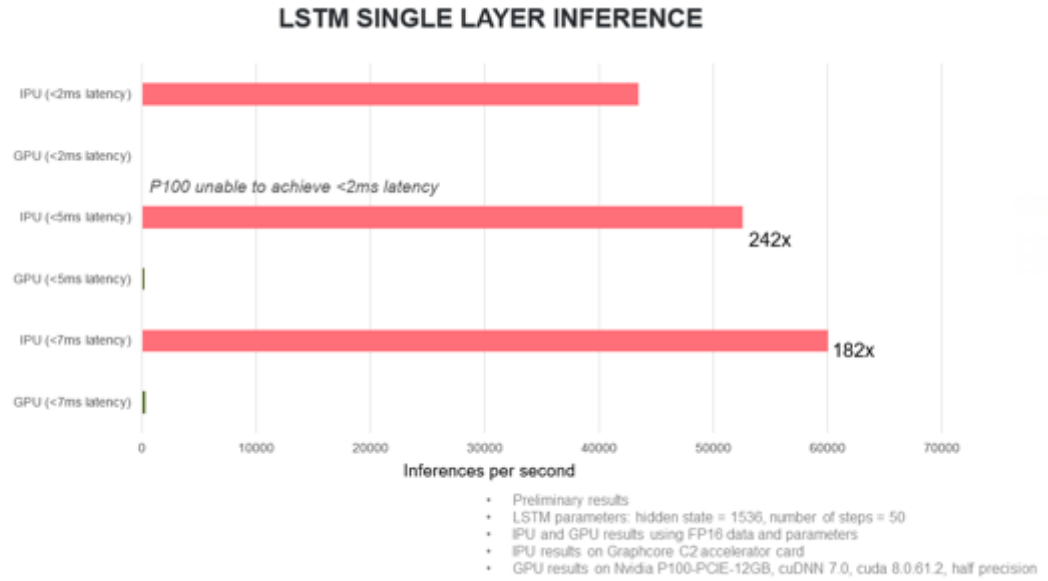


Courtesy: https://ai.intel.com/intel-nervana-neural-network-processor-architecture-update/

GraphCore – Intelligence Processing Unit (IPU)

- New processor that's the first to be specifically designed for machine intelligence workloads – an Intelligence Processing Unit (IPU)
  - Massively parallel
  - Low-precision floating-point compute
  - Higher compute density
- UK-based Startup
- Early benchmarks show 10-100x speedup over GPUs
  - Presented at NIPS 2017



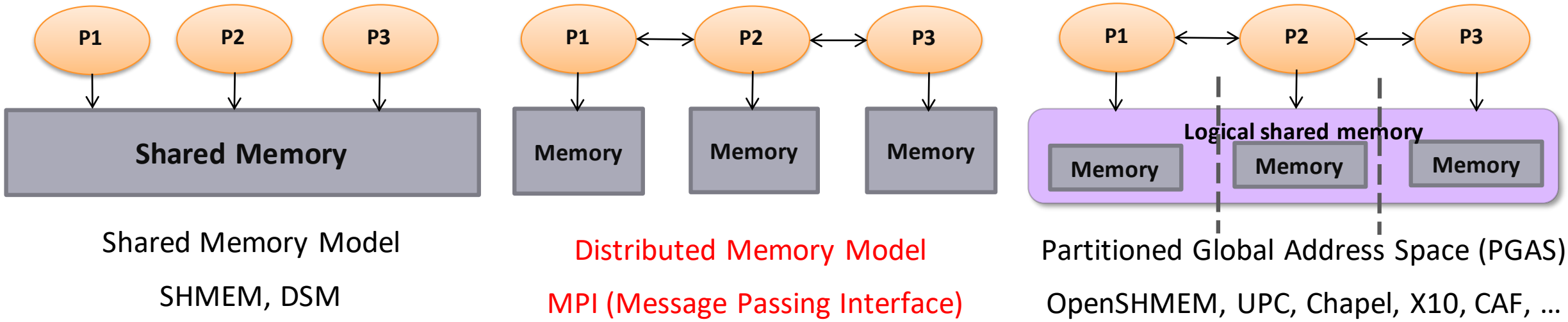


Courtesy: https://www.graphcore.ai/posts/preliminary-ipu-benchmarks-providing-previously-unseen-performance-for-a-range-of-machine-learning-applications

HPC Technologies

- Hardware
  - Interconnects – InfiniBand, RoCE, Omni-Path, etc.
  - Processors – GPUs, Multi-/Many-core CPUs, Tensor Processing Unit (TPU), FPGAs, etc.
- **Communication Middleware**
  - **Message Passing Interface (MPI)**
    - **CUDA-Aware MPI, Many-core Optimized MPI runtimes (KNL-specific optimizations)**
  - **NVIDIA NCCL**

Parallel Programming Models Overview



Shared Memory Model

SHMEM, DSM

Distributed Memory Model

MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)

OpenSHMEM, UPC, Chapel, X10, CAF, …

- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- PGAS models and Hybrid MPI+PGAS models are gradually receiving importance

Allreduce Collective Communication Pattern

- Element-wise Sum data from all processes and sends to all processes

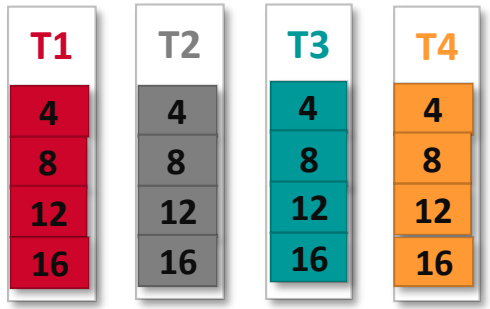
int MPI_Allreduce (const void *sendbuf, void * recvbuf, int count, MPI_Datatype datatype, MPI_Op operation, MPI_Comm comm)

Input-only Parameters	
Parameter	**Description**
sendbuf	Starting address of send buffer
recvbuf	Starting address of recv buffer
type	Data type of buffer elements
count	Number of elements in the buffers
operation	Reduction operation to be performed (e.g. sum)
comm	Communicator handle

Input/Output Parameters	
Parameter	**Description**
recvbuf	Starting address of receive buffer

Sendbuf (Before)

Recvbuf (After)



Overview of the MVAPICH2 Project

- High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.1), Started in 2001, First version available in 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2011
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  - Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
  - **Used by more than 2,975 organizations in 86 countries**
  - **More than 528,000 (> 0.5 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (Nov '18 ranking)
    - 3rd ranked 10,649,640-core cluster (Sunway TaihuLight) at NSC, Wuxi, China
    - 14th, 556,104 cores (Oakforest-PACS) in Japan
    - 17th, 367,024 cores (Stampede2) at TACC
    - 27th, 241,108-core (Pleiades) at NASA and many others
  - Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, and OpenHPC)
  - http://mvapich.cse.ohio-state.edu

Partner in the upcoming TACC Frontera System

- Empowering Top500 systems for over a decade

GPU-Aware (CUDA-Aware) MPI Library: MVAPICH2-GDR

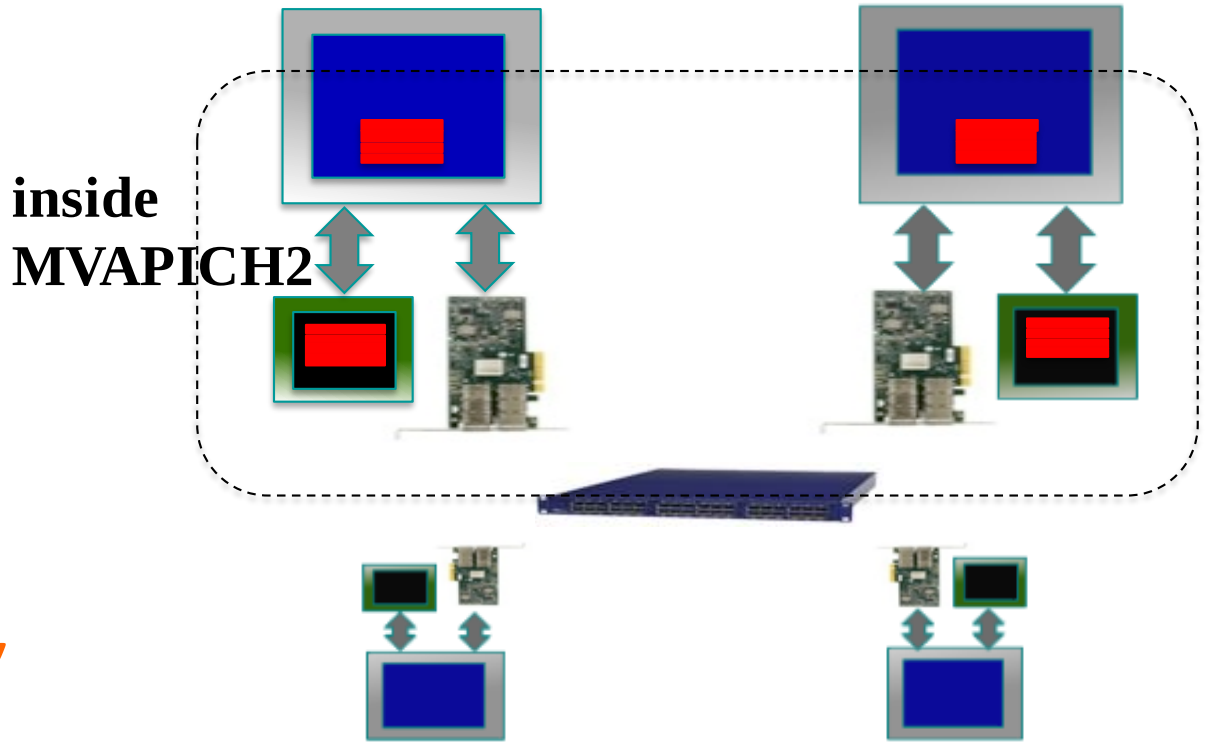
- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, …);
```

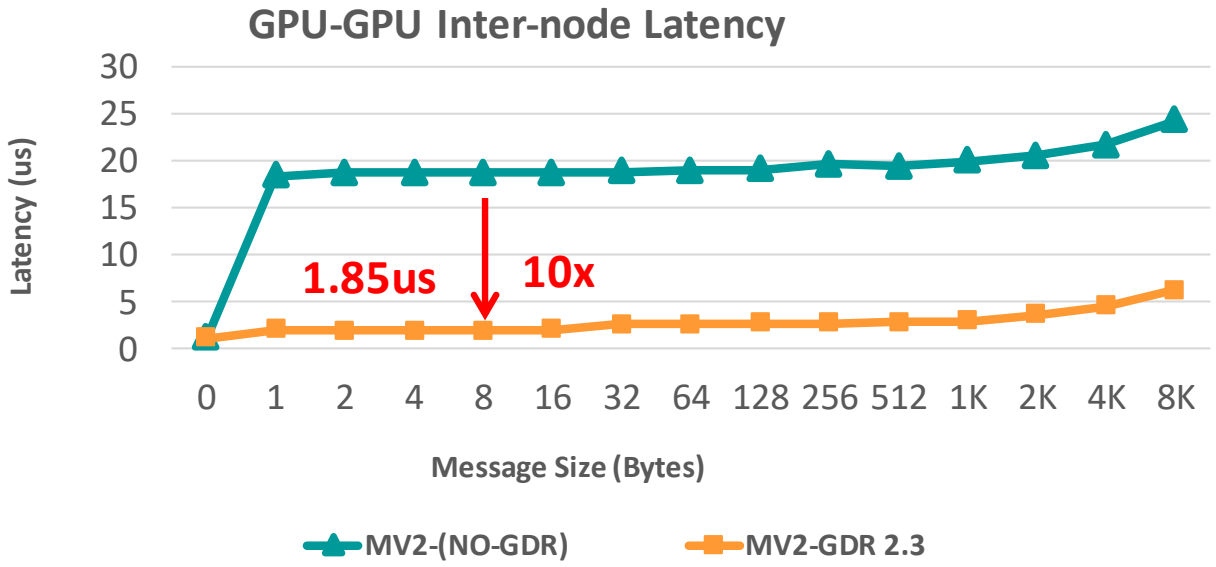
At Receiver:

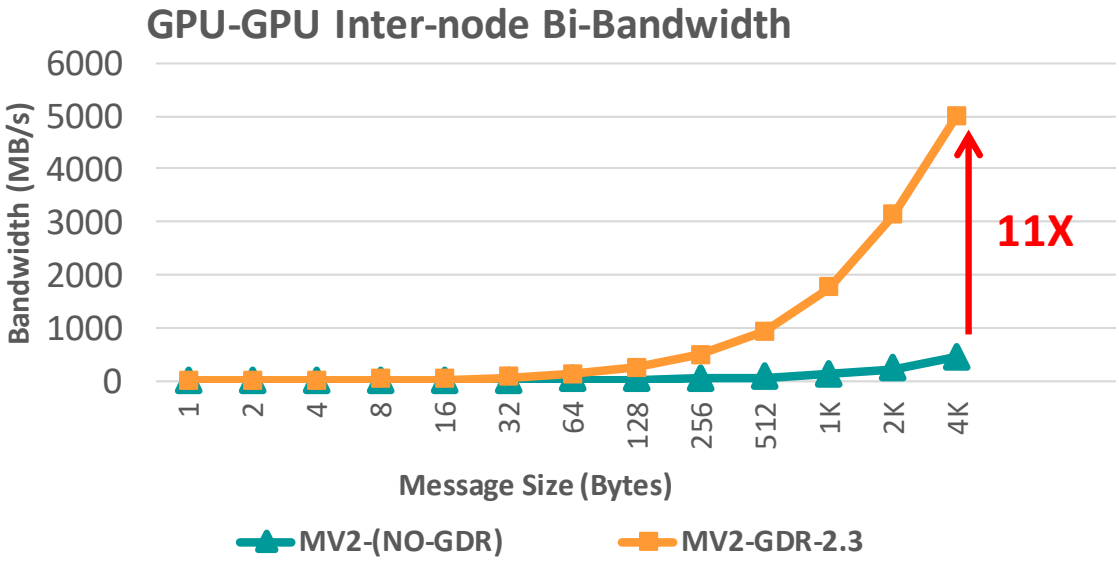
```
MPI_Recv(r_devbuf, size, …);
```

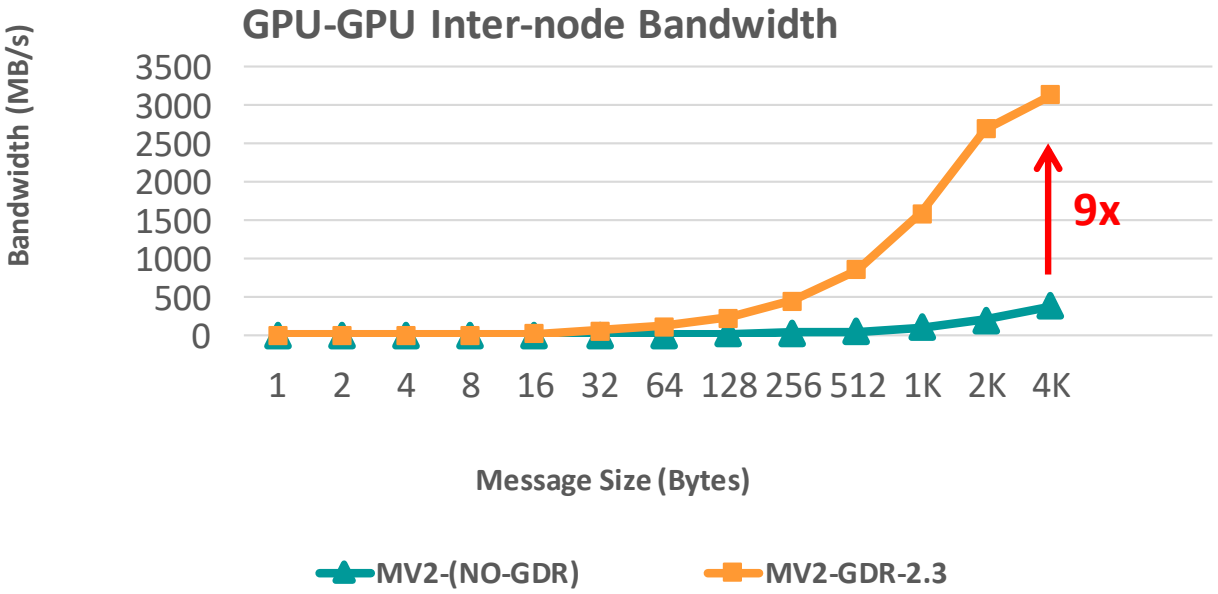


High Performance and High Productivity

Optimized MVAPICH2-GDR Design



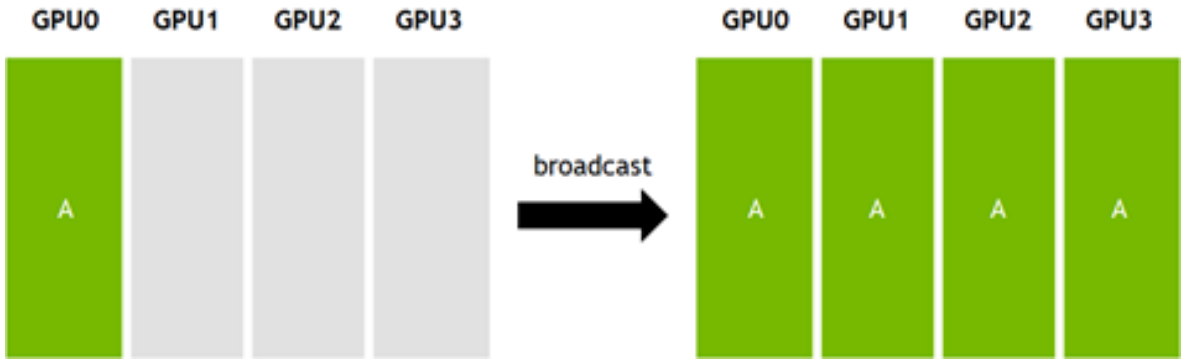


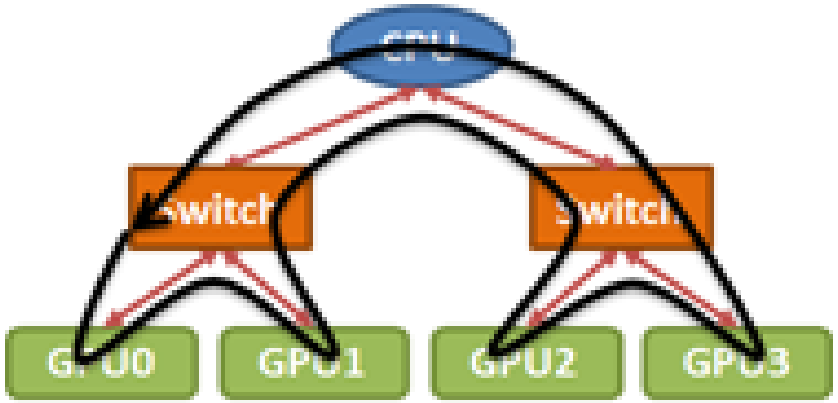


MVAPICH2-GDR-2.3.1
Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores
NVIDIA Volta V100 GPU
Mellanox Connect-X4 EDR HCA
CUDA 9.0
Mellanox OFED 4.0 with GPU-Direct-RDMA

NCCL Communication Library

- Collective Communication with a caveat!
  - GPU buffer exchange
  - **Dense Multi-GPU** systems (Cray CS-Storm, DGX-1)
  - MPI-like – but not MPI standard compliant
- NCCL (pronounced Nickel)
  - Open-source Communication Library by NVIDIA
  - Topology-aware, ring-based (linear) collective communication library for GPUs
  - Divide bigger buffers to smaller chunks
  - Good performance for large messages
    - Kernel-based threaded copy (Warp-level Parallel) instead of cudaMemcpy





https://devblogs.nvidia.com/parallelforall/fast-multi-gpu-collectives-nccl/

Outline

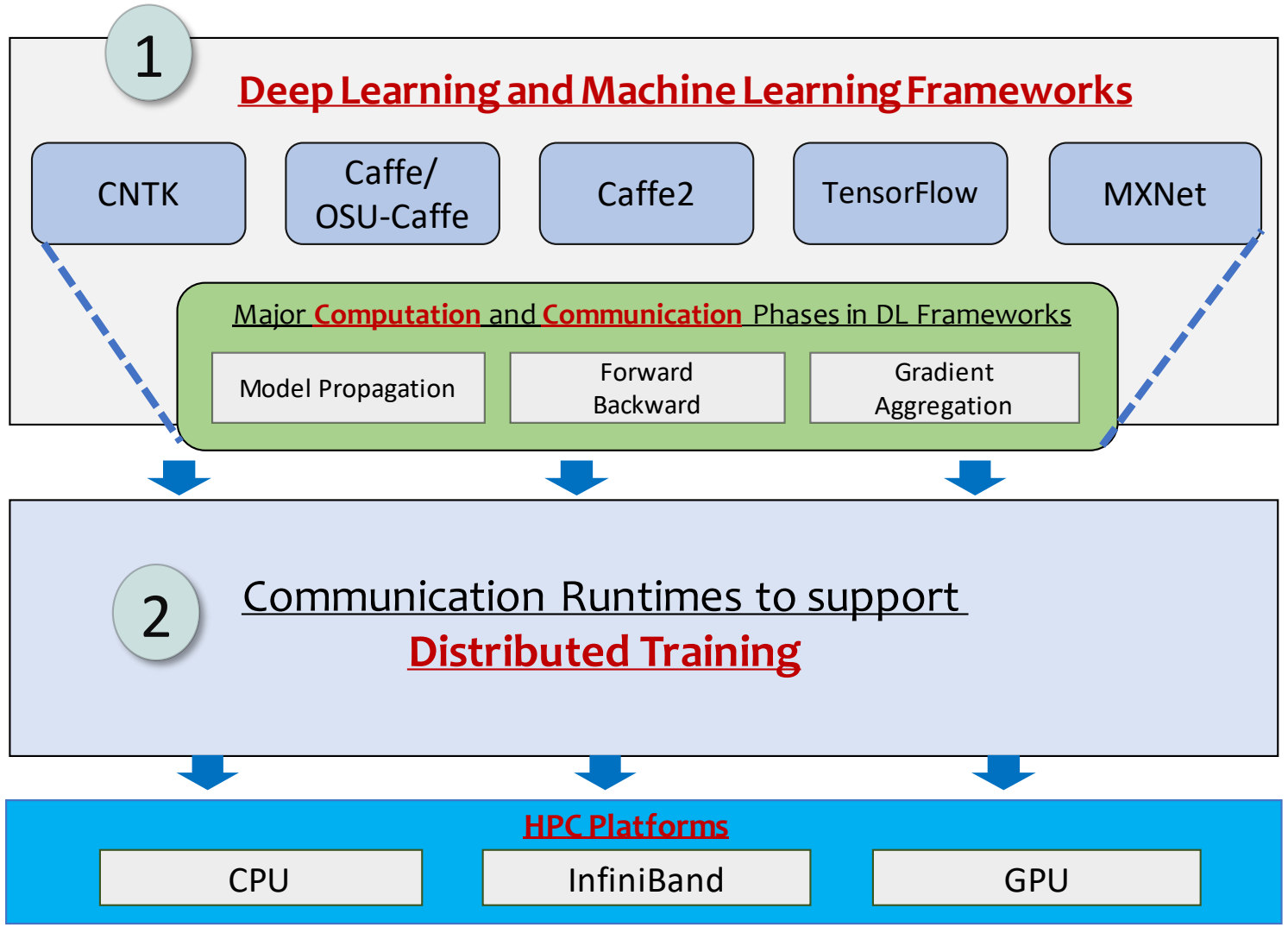
- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- **Challenges in Exploiting HPC Technologies for Deep Learning**
- Solutions and Case Studies
- Open Issues and Challenges
- Conclusion

Broad Challenge: Exploiting HPC for Deep Learning

How to efficiently scale-out a Deep Learning (DL) framework and take advantage of heterogeneous High Performance Computing (HPC) resources?

Research Challenges to Exploit HPC Technologies

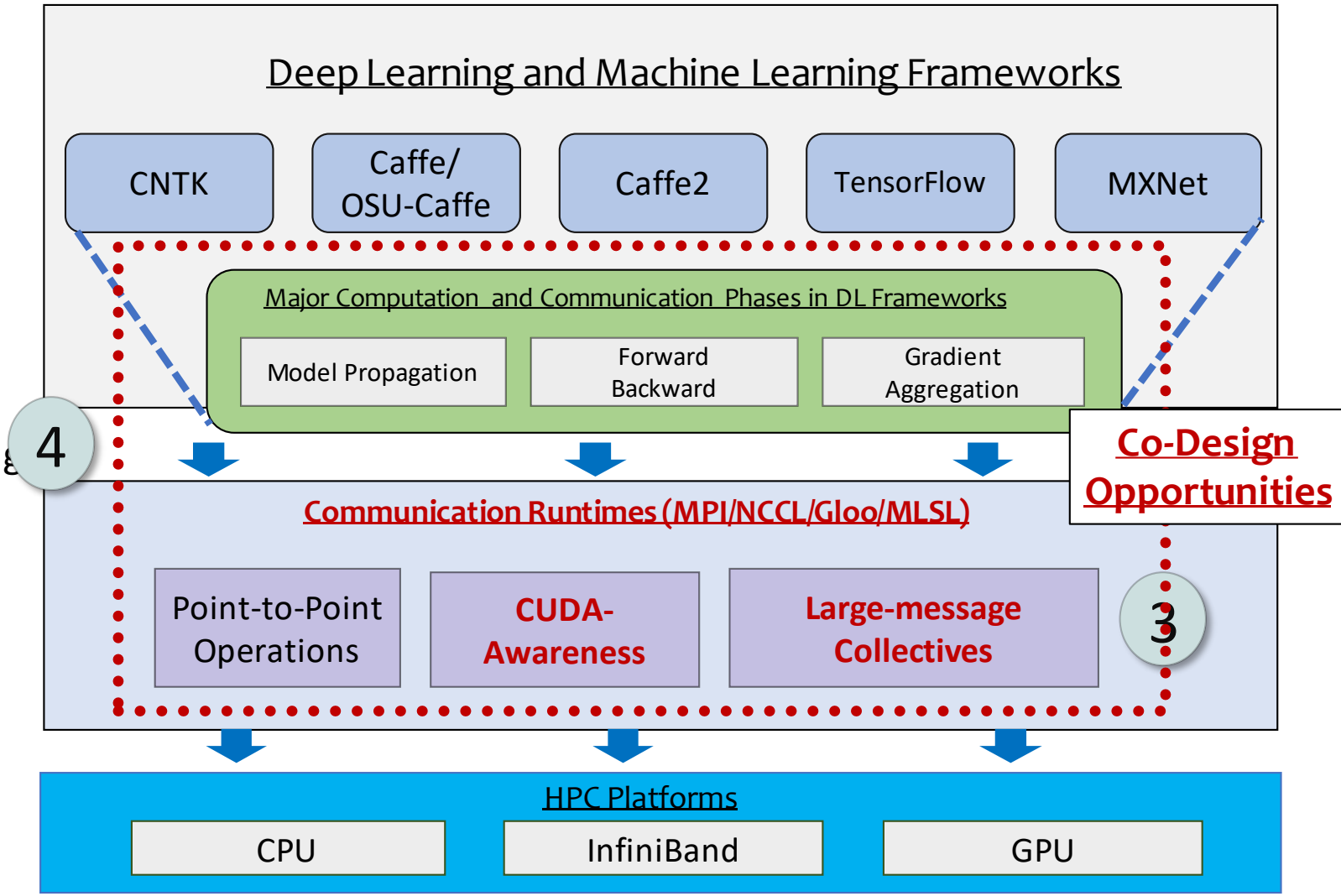
1. What are the fundamental issues in designing **DL frameworks**?
   - Memory Requirements
   - **Computation** Requirements
   - **Communication** Overhead
2. Why do we need to support **distributed training**?
   - To overcome the limits of single-node training
   - To better utilize hundreds of existing HPC Clusters



Research Challenges to Exploit HPC Technologies (Cont'd)

3. What are the **new design challenges** brought forward by DL frameworks for Communication runtimes?
   - Large Message **Collective Communication** and Reductions
   - GPU Buffers (**CUDA-Awareness**)

4. Can a **Co-design** approach help in achieving Scale-up and Scale-out efficiently?
   - **Co-Design** the support at **Runtime level** and Exploit it at the **DL Framework level**
   - What performance benefits can be observed?
   - What needs to be fixed at the **communication runtime** layer?

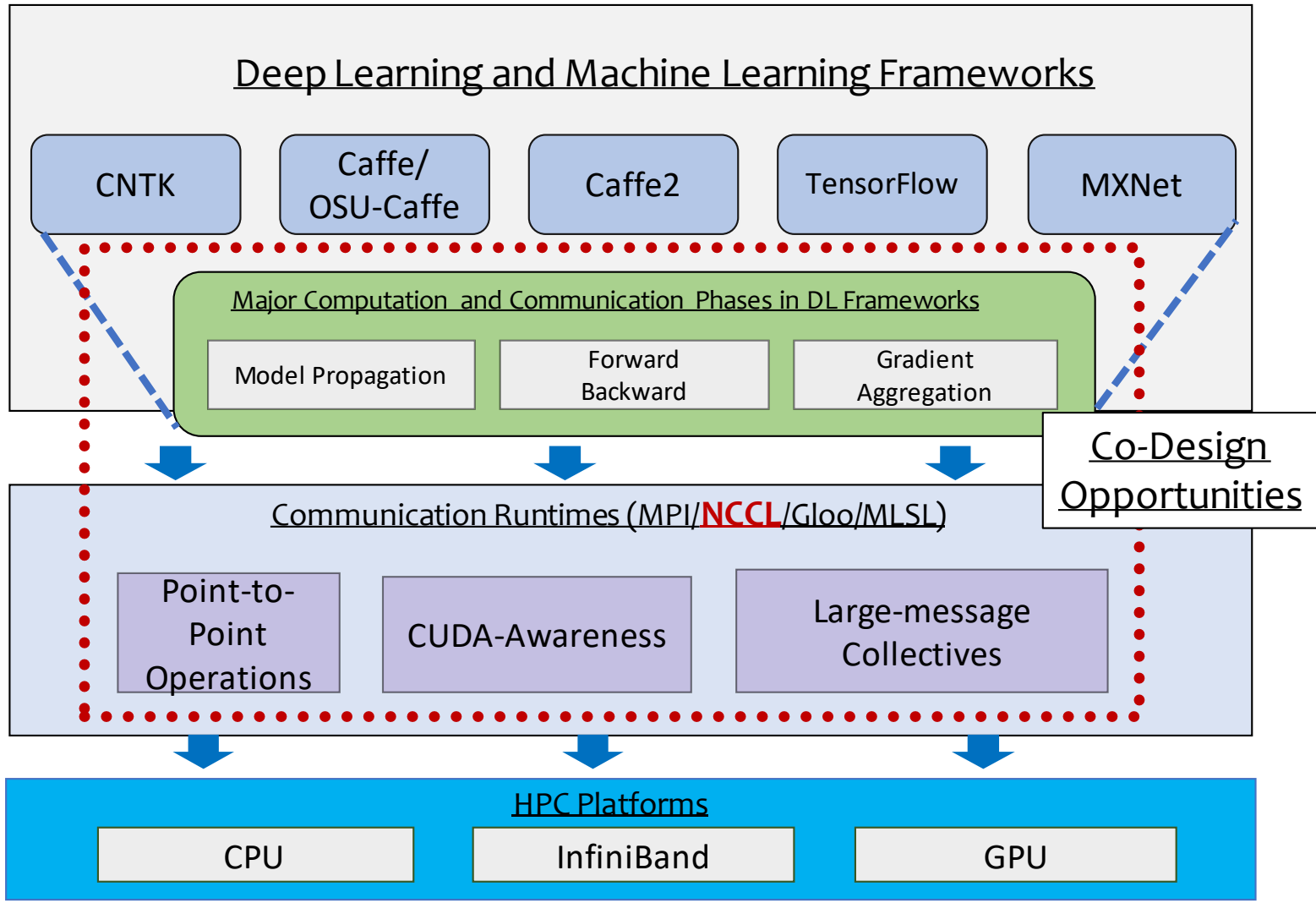


Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- **Solutions and Case Studies**
- Open Issues and Challenges
- Conclusion

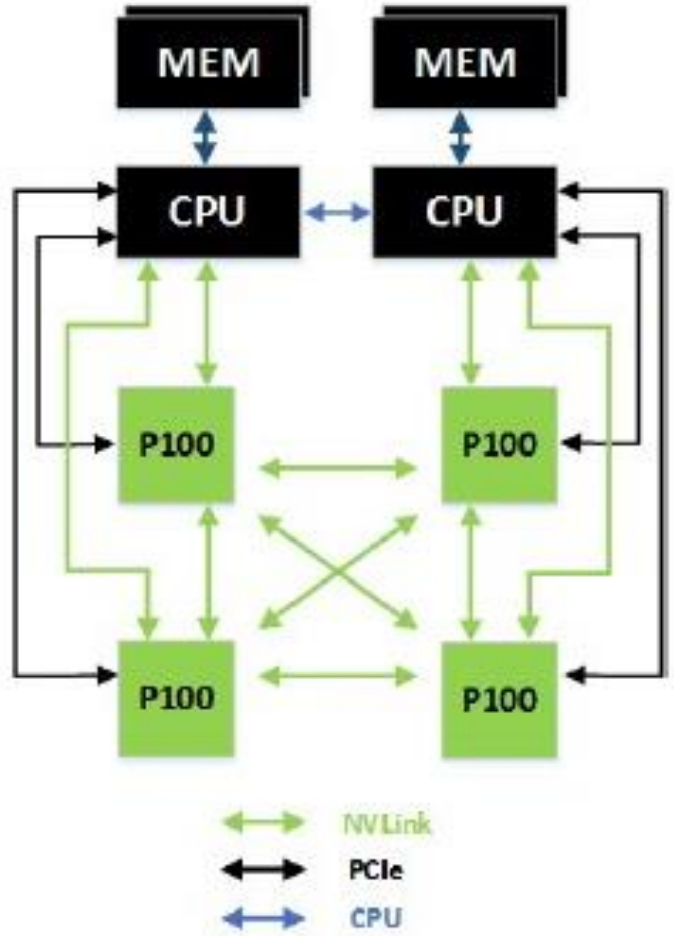
Solutions and Case Studies: Exploiting HPC for DL

- **NVIDIA NCCL/NCCL2**
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL



NVIDIA NCCL

- NCCL is a collective communication library
  - NCCL 1.x is only for Intra-node communication on a single-node
- NCCL 2.0 supports inter-node communication as well
- Design Philosophy
  - Use Rings and CUDA Kernels to perform efficient communication
- NCCL is optimized for dense multi-GPU systems like the DGX-1 and DGX-1V



Fully connected quad

120 GB/s per GPU bidirectional for peer traffic

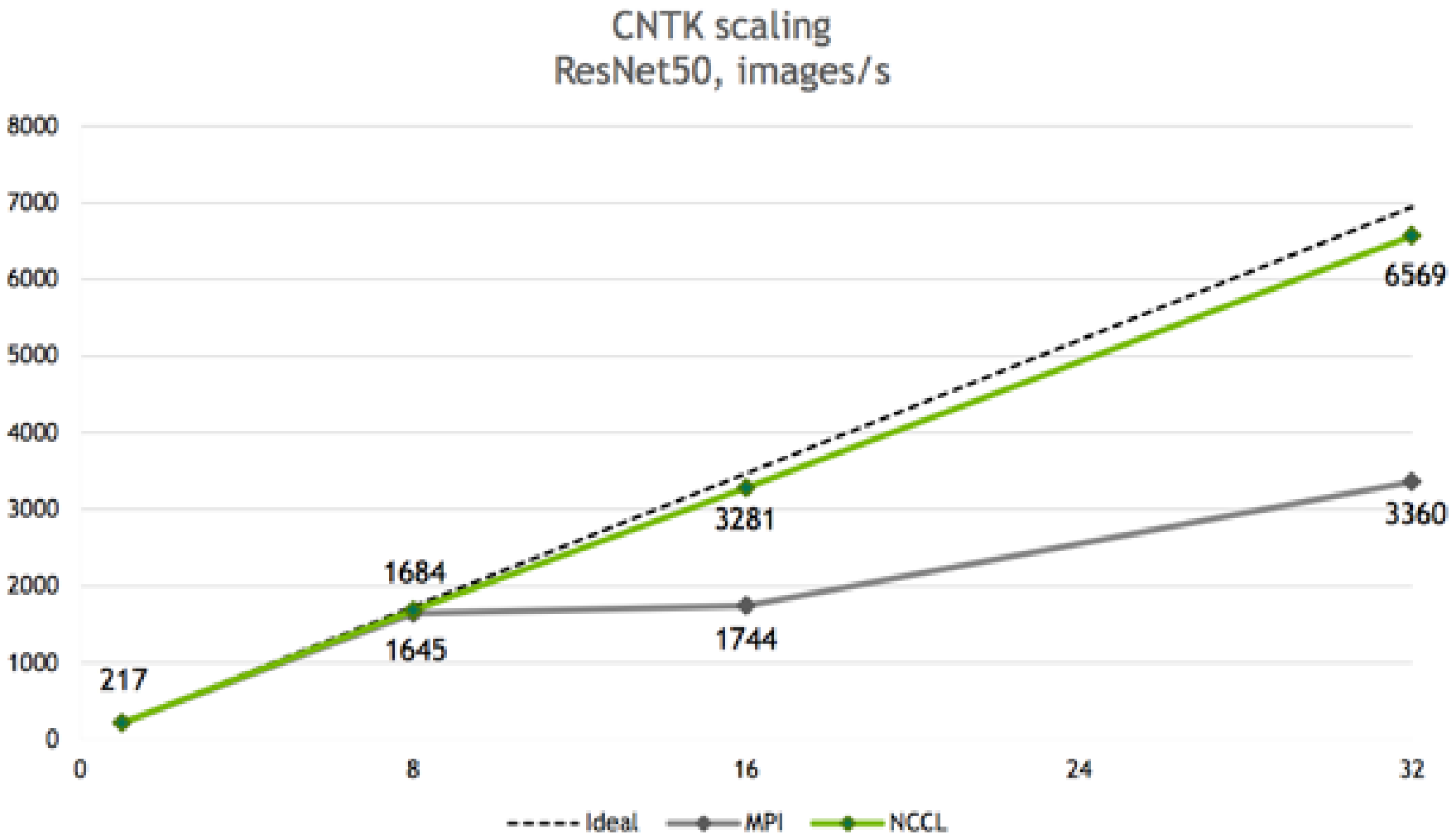
40 GB/s per GPU bidirectional to CPU

Direct Load/store access to CPU Memory

High Speed Copy Engines for bulk data movement

Courtesy: https://www.nextplatform.com/2016/05/04/nvlink-takes-gpu-acceleration-next-level/

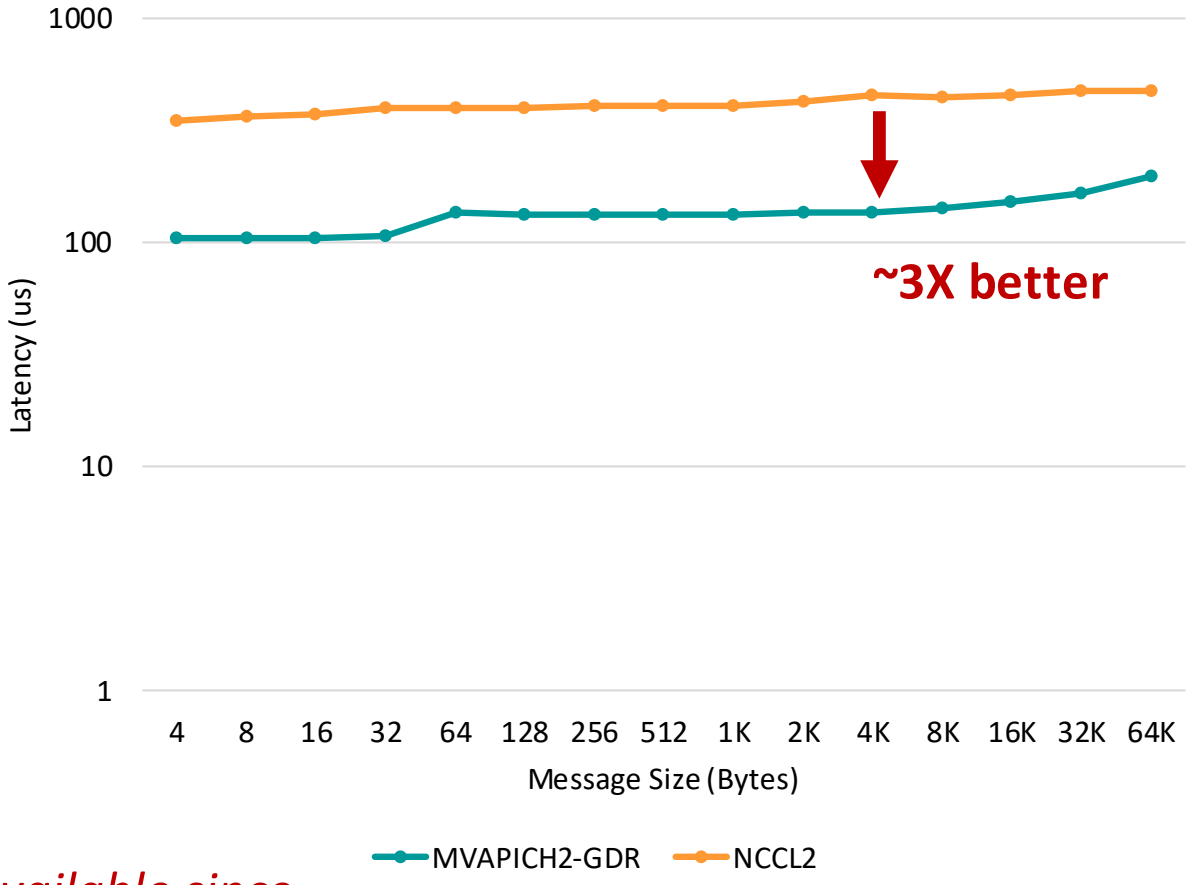
NCCL 2: Multi-node GPU Collectives

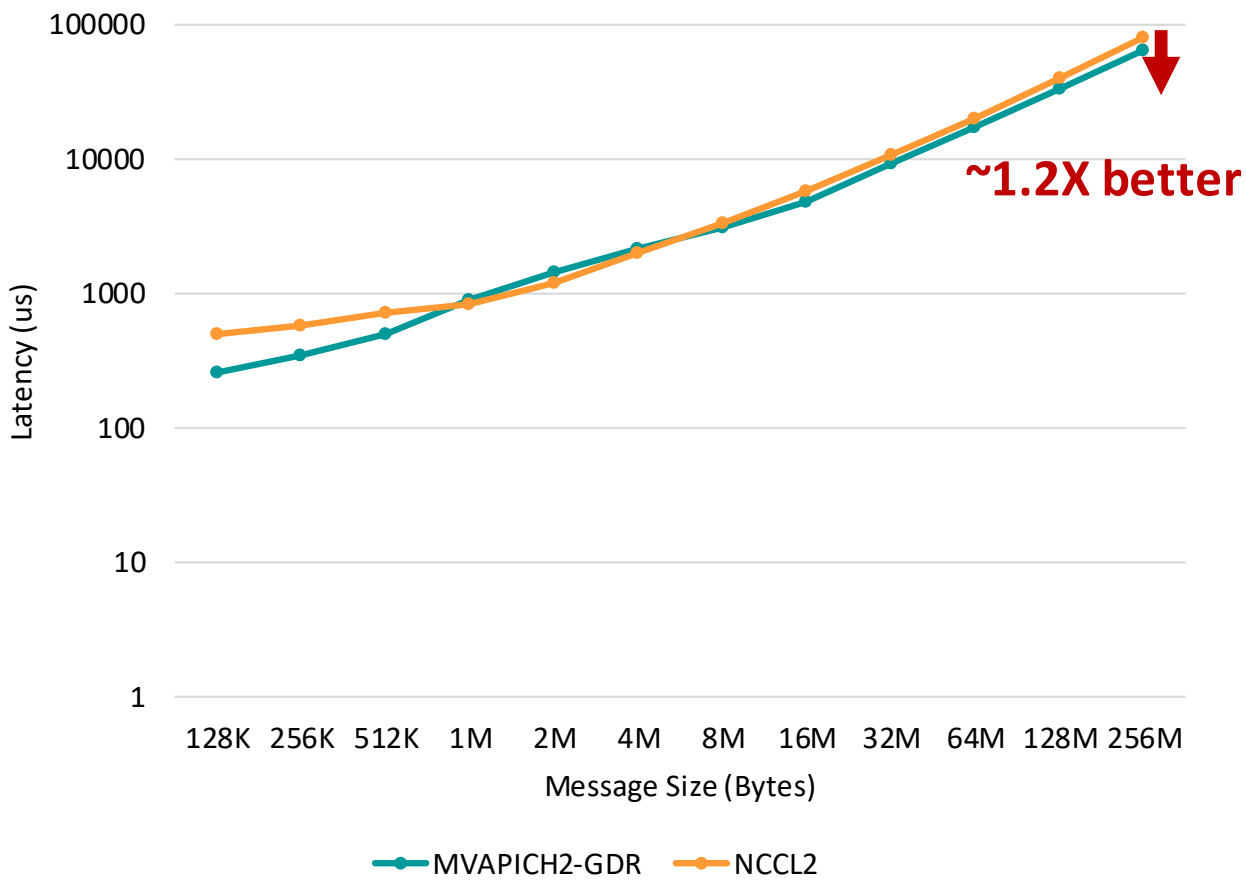


Courtesy: http://on-demand.gputechconf.com/gtc/2017/presentation/s7155-jeaugey-nccl.pdf

MVAPICH2-GDR vs. NCCL2 – Allreduce Operation

- Optimized designs in MVAPICH2-GDR 2.3 offer better/comparable performance for most cases
- MPI_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) on 16 GPUs



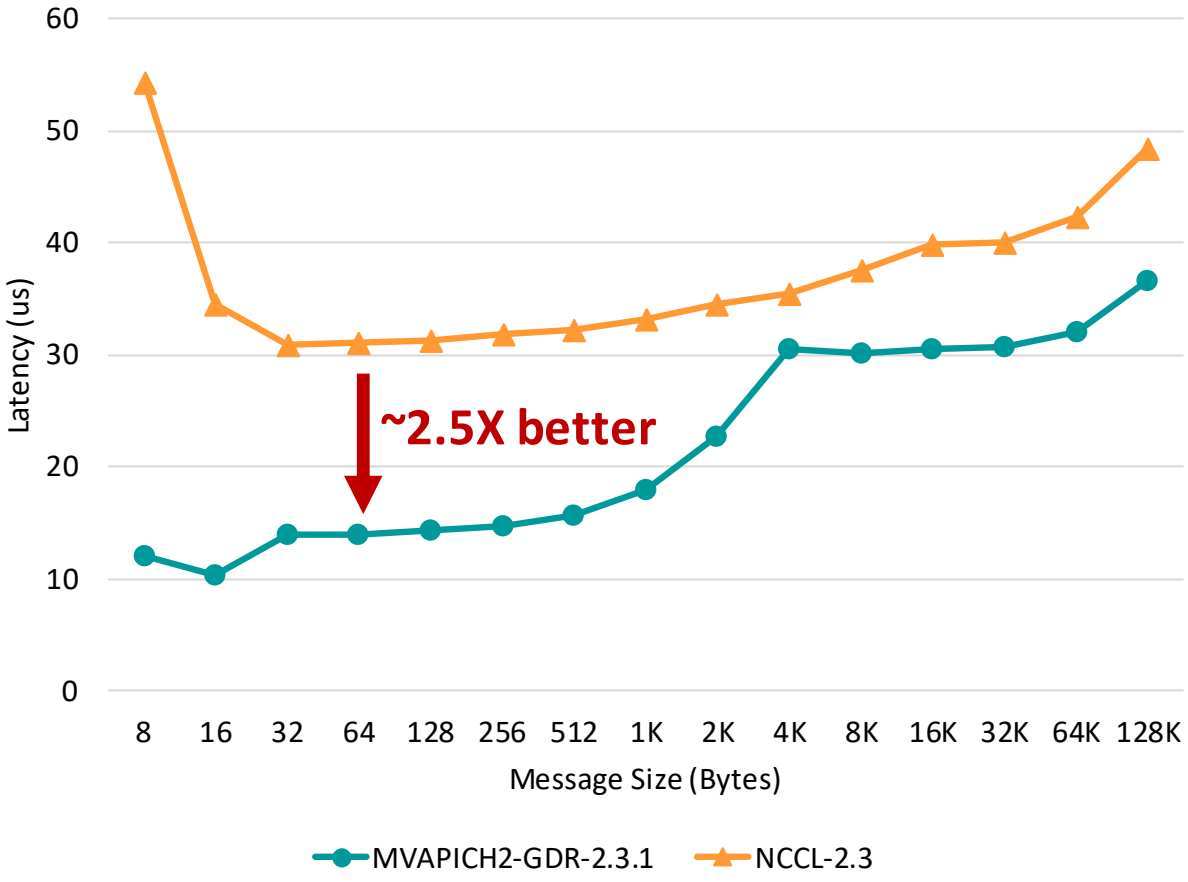


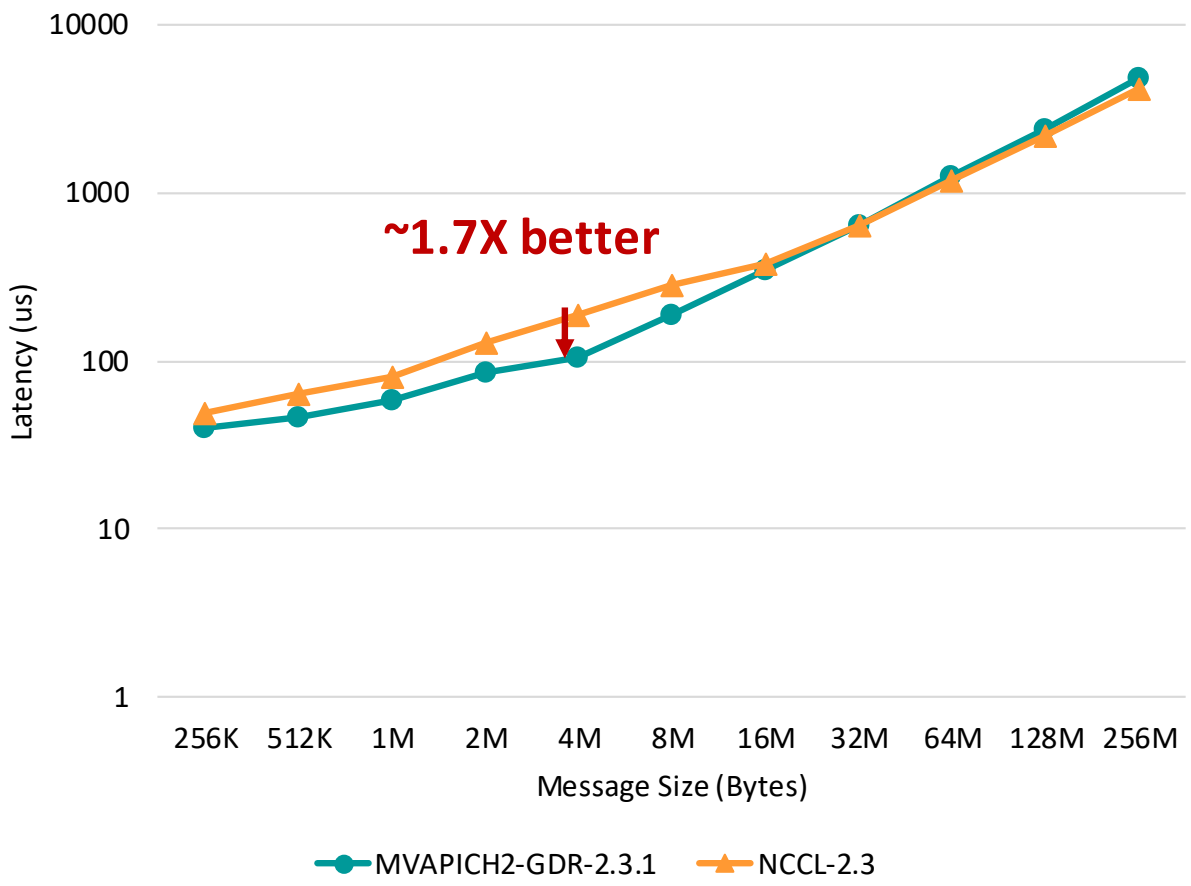
*Available since MVAPICH2-GDR 2.3

Platform: Intel Xeon (Broadwell) nodes equipped with a dual-socket CPU, 1 K-80 GPUs, and EDR InfiniBand Inter-connect

MVAPICH2-GDR vs. NCCL2 – Allreduce on DGX-2

- Optimized designs in MVAPICH2-GDR 2.3.1 offer better/comparable performance for most cases
- MPI_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) on 1 DGX-2 node (16 Volta GPUs)



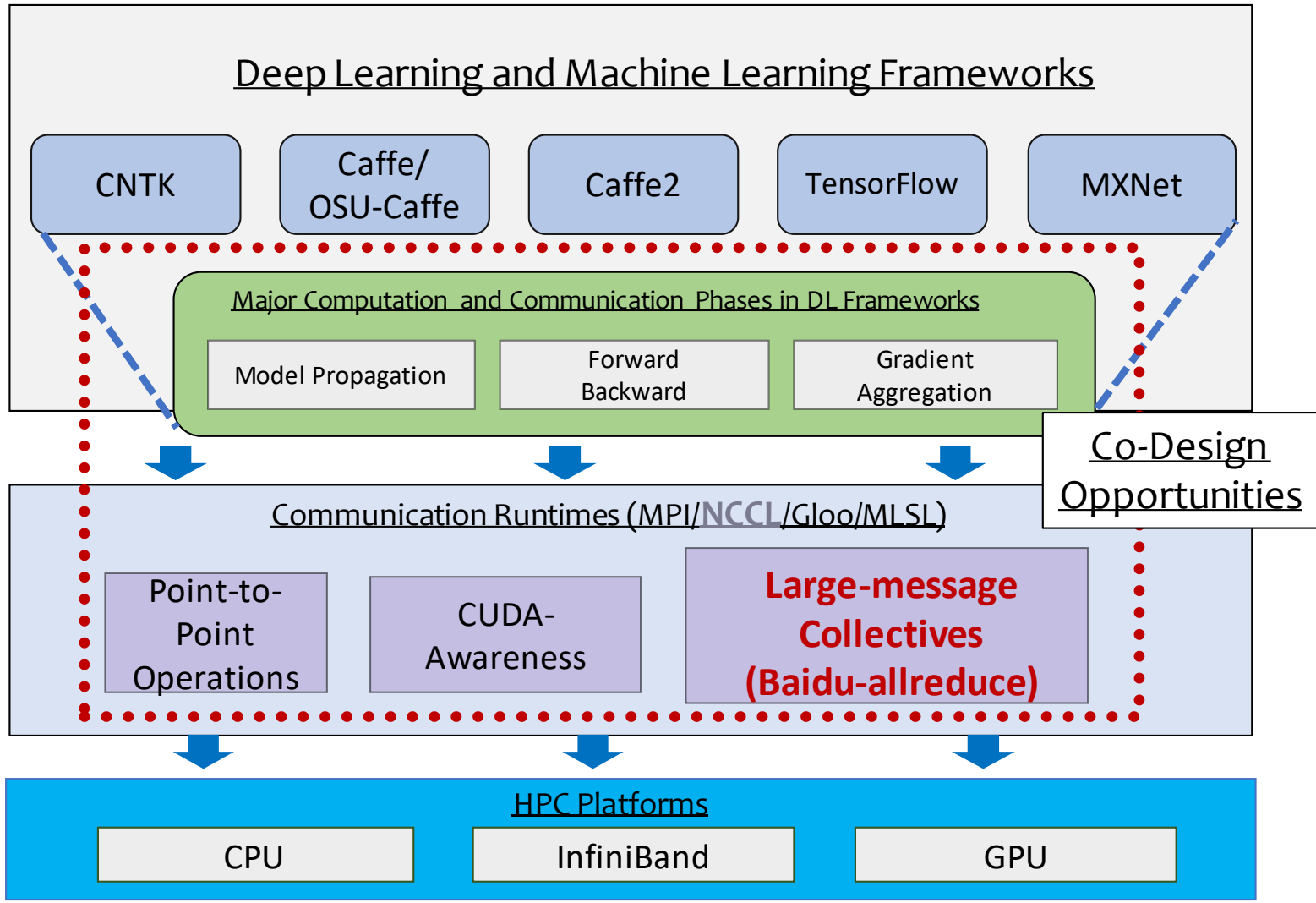


**Available with MVAPICH2-GDR 2.3.1*

Platform: Nvidia DGX-2 system (16 Nvidia Volta GPUs connected with NVSwitch), CUDA 9.2

Solutions and Case Studies: Exploiting HPC for DL

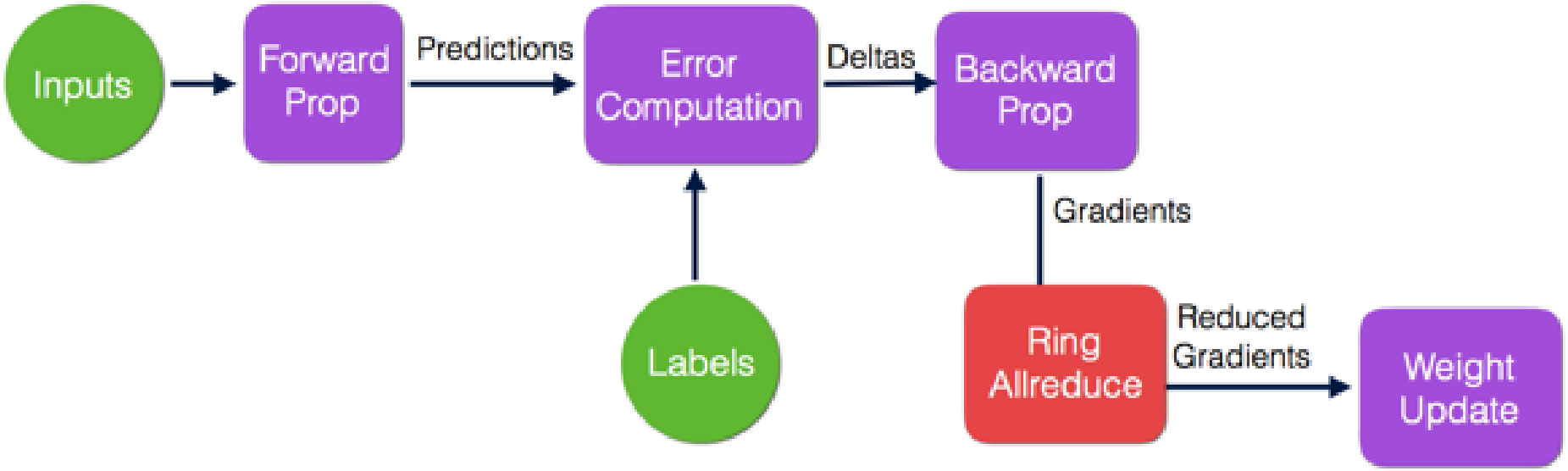
- NVIDIA NCCL
- **Baidu-allreduce**
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL



Baidu's Ring-Allreduce in TensorFlow

Scaling with TensorFlow

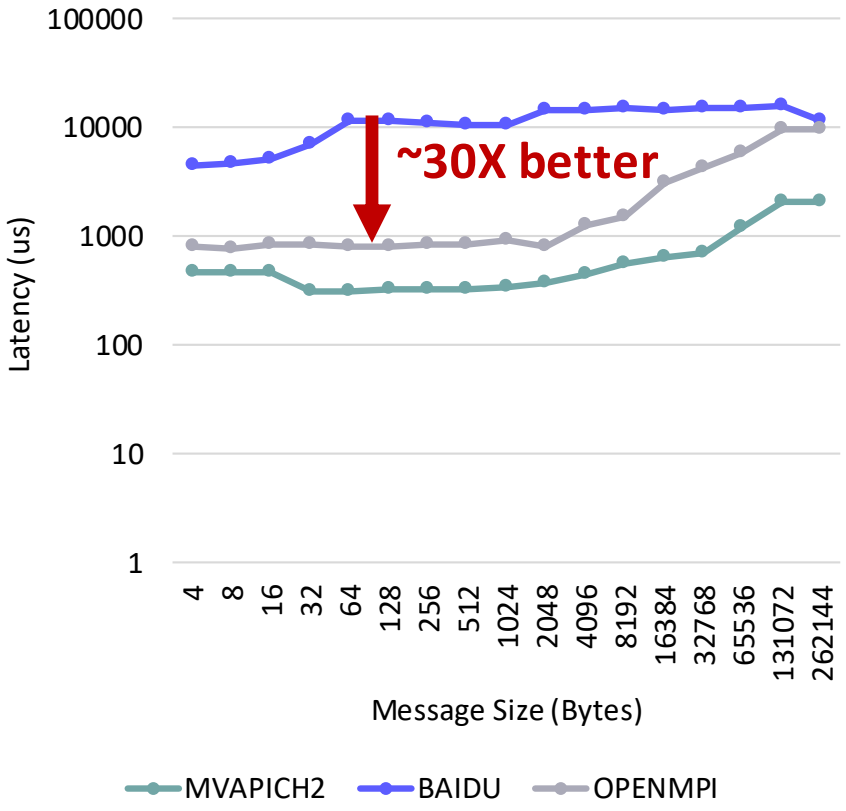
- Run many independent TensorFlow processes
- Insert allreduce as a node in the graph:

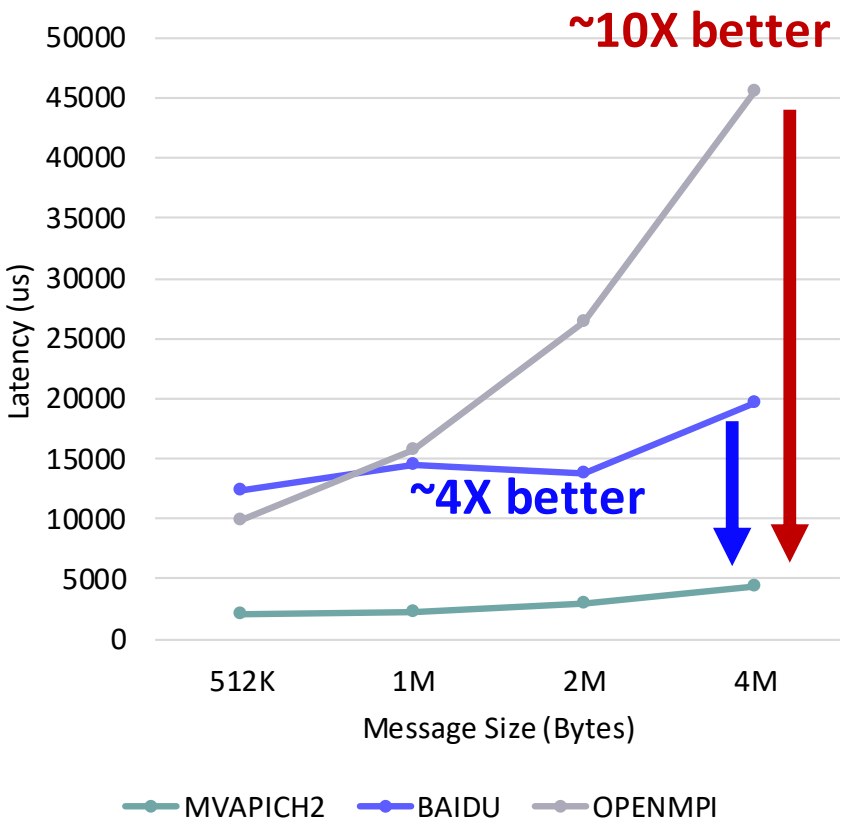


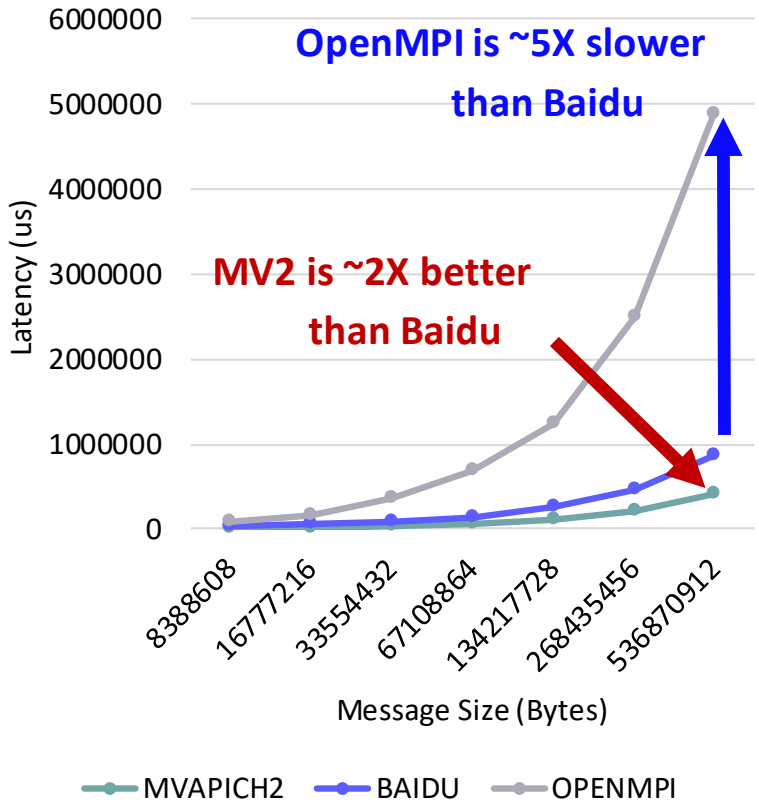
Courtesy: http://on-demand.gputechconf.com/gtc/2017/presentation/s7543-andrew-gibiansky-effectively-scakukbg-deep-learning-frameworks.pdf

MVAPICH2-GDR: Allreduce Comparison with Baidu and OpenMPI

- 16 GPUs (4 nodes) MVAPICH2-GDR vs. Baidu-Allreduce and OpenMPI 3.0



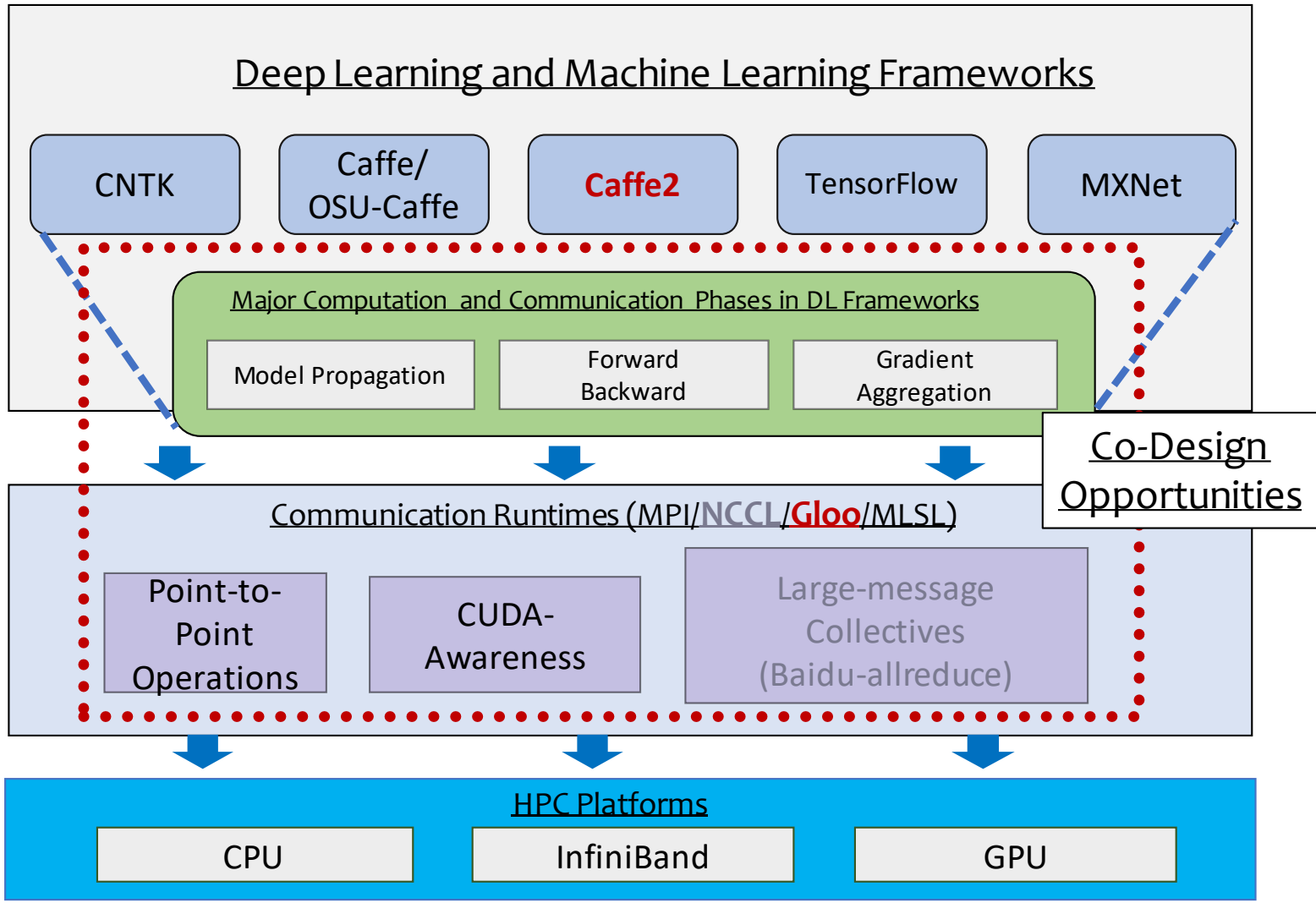




**Available since MVAPICH2-GDR 2.3a*

Solutions and Case Studies: Exploiting HPC for DL

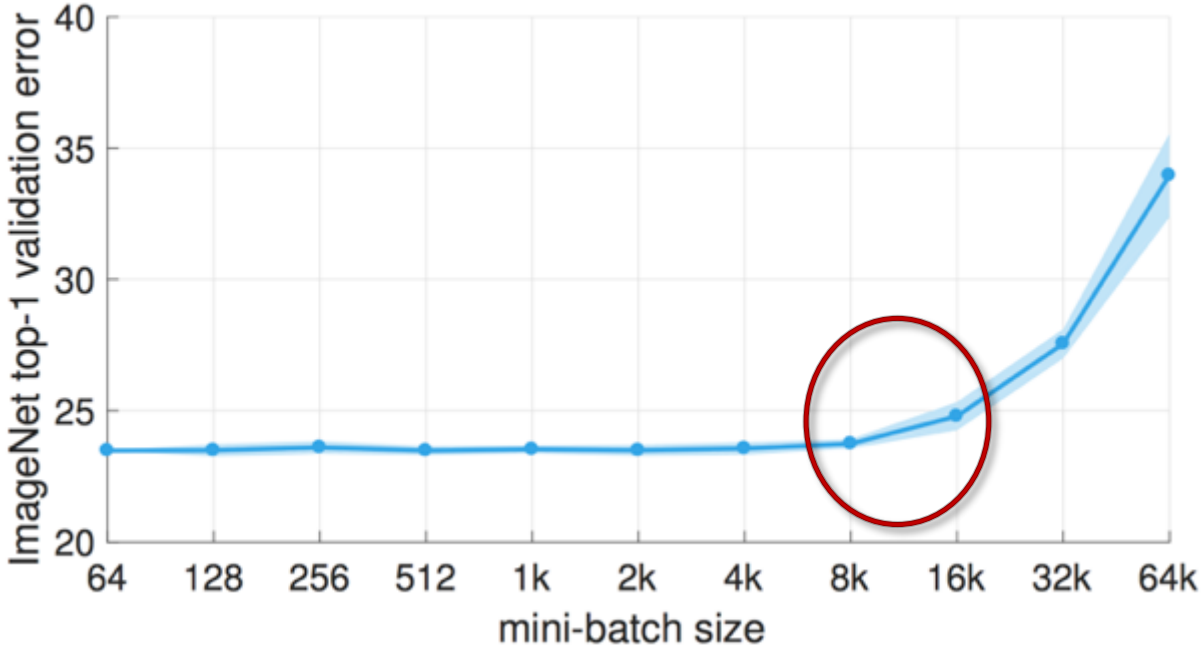
- NVIDIA NCCL
- Baidu-allreduce
- **Facebook Gloo**
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL

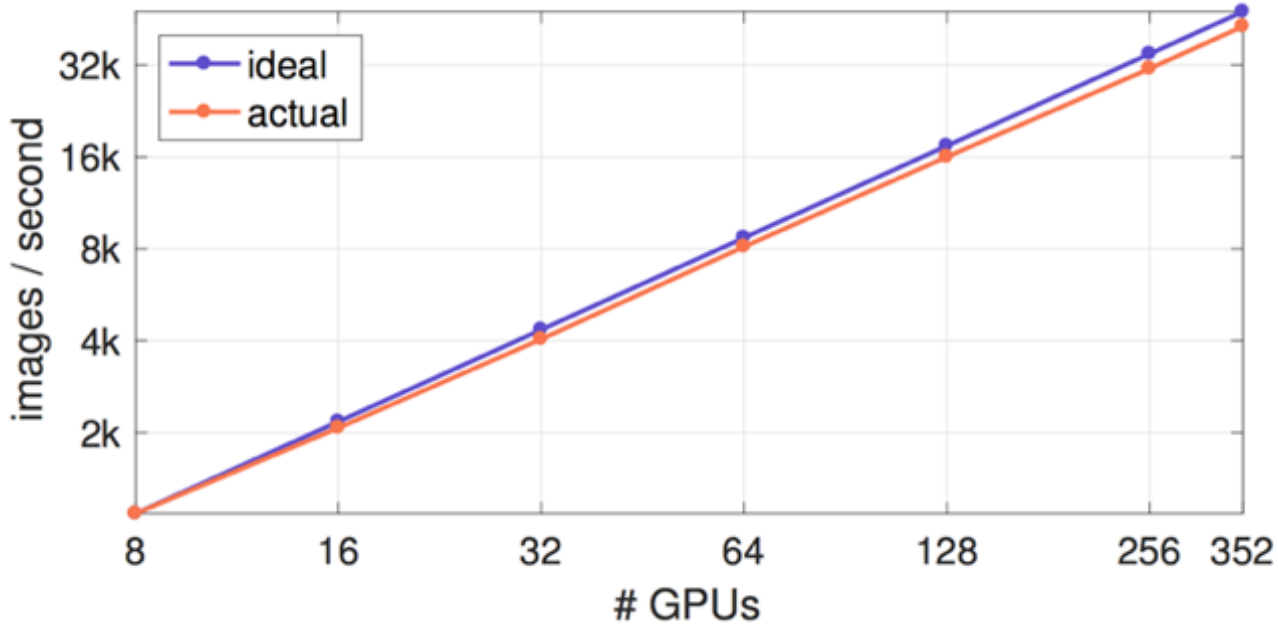


Facebook Caffe2

- Caffe2 (by Facebook) allows the use of multiple communication back-ends
  - Gloo – Multi-node design from the beginning
  - NCCL – Multi-node support added recently in v2
- Gloo – Performance evaluation studies not available yet
- Design principles are similar to MPI and NCCL
- In essence, Gloo is an application level implementation of collective algorithms for Reduce, Allreduce, etc.
- Details and code available from: https://github.com/facebookincubator/gloo

Facebook: Training ImageNet in 1 Hour



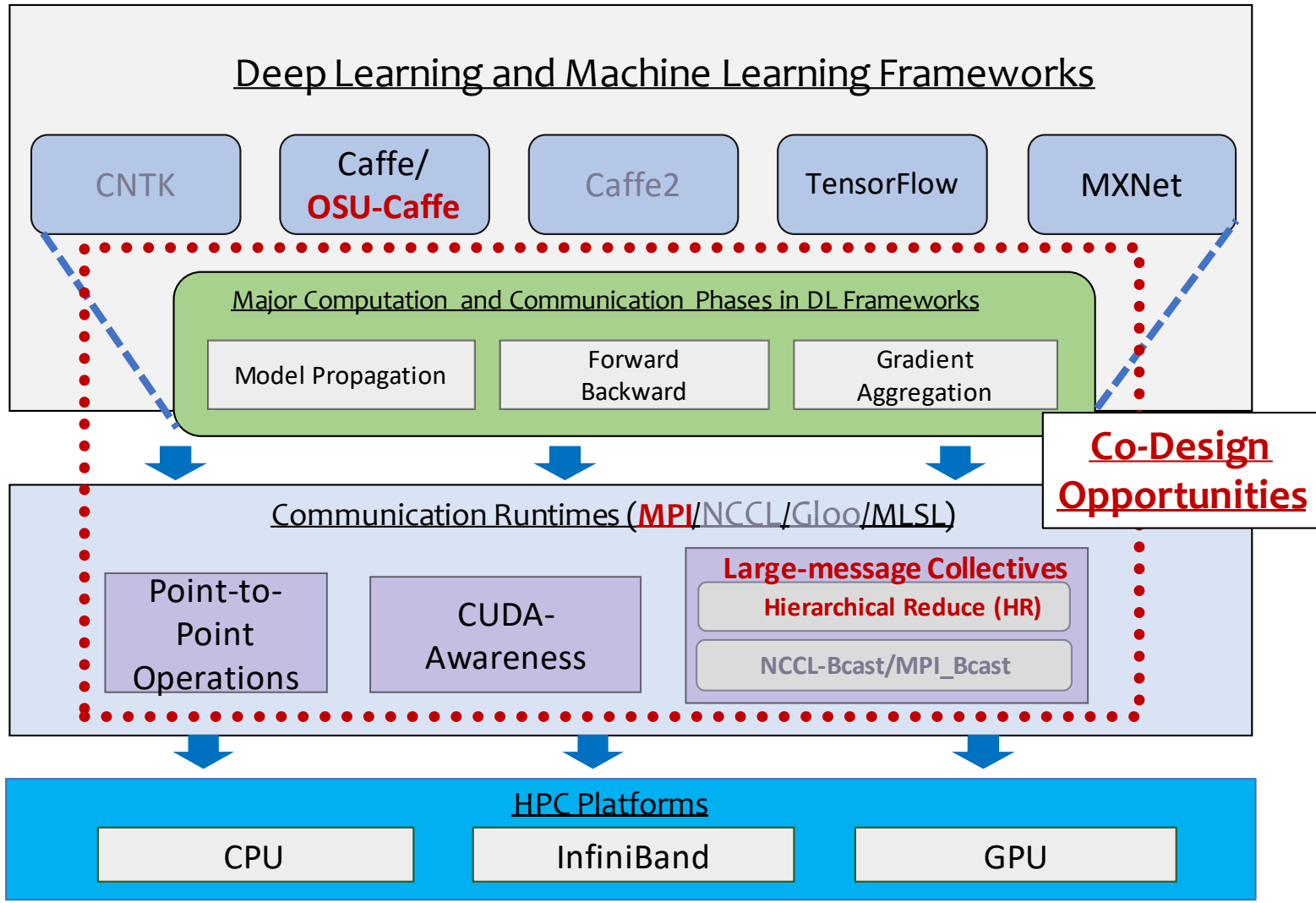


- Near-linear Scaling for ~256 Pascal GPUs (Facebook Big Basin Servers with 8 GPUs/node)
- Explored large batch-size training with ResNet-50
  - ***8K batch-size seems to be the sweet-spot.***

Courtesy: https://research.fb.com/publications/imagenet1kin1h/

Solutions and Case Studies: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce
- Facebook Gloo
- **Co-design MPI runtimes and DL Frameworks**
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL



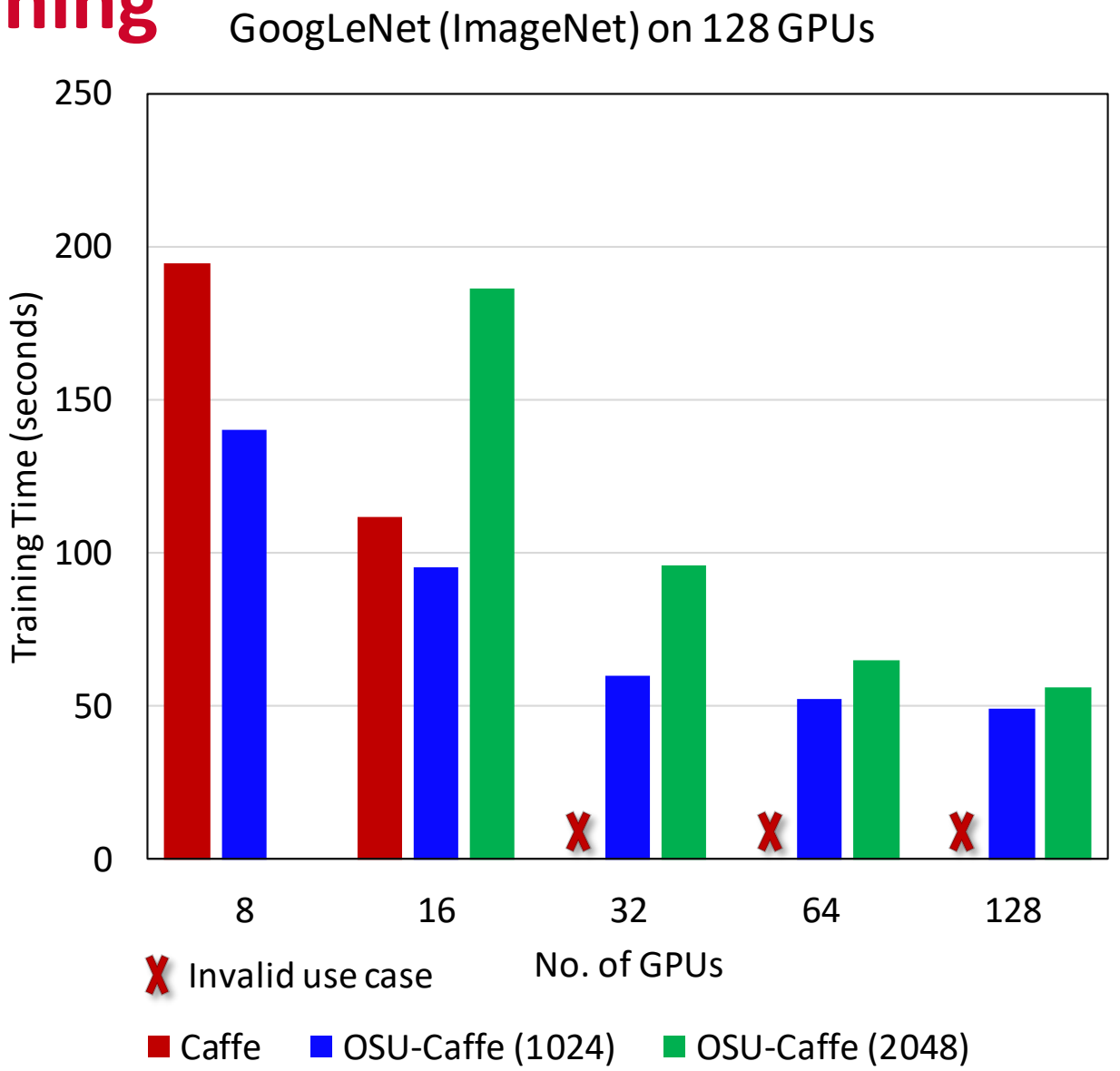
S-Caffe: Proposed Co-Design Overview

- To address the limitations of Caffe and existing MPI runtimes, we propose the **OSU-Caffe (S-Caffe)** framework
- At the application (DL framework) level
  - Develop a fine-grain workflow – i.e. layer-wise communication instead of communicating the entire model
- At the runtime (MPI) level
  - Develop support to perform reduction of very-large GPU buffers
  - Perform reduction using GPU kernels

OSU-Caffe is available from the HiDL project page (http://hidl.cse.ohio-state.edu)

OSU-Caffe: Scalable Deep Learning

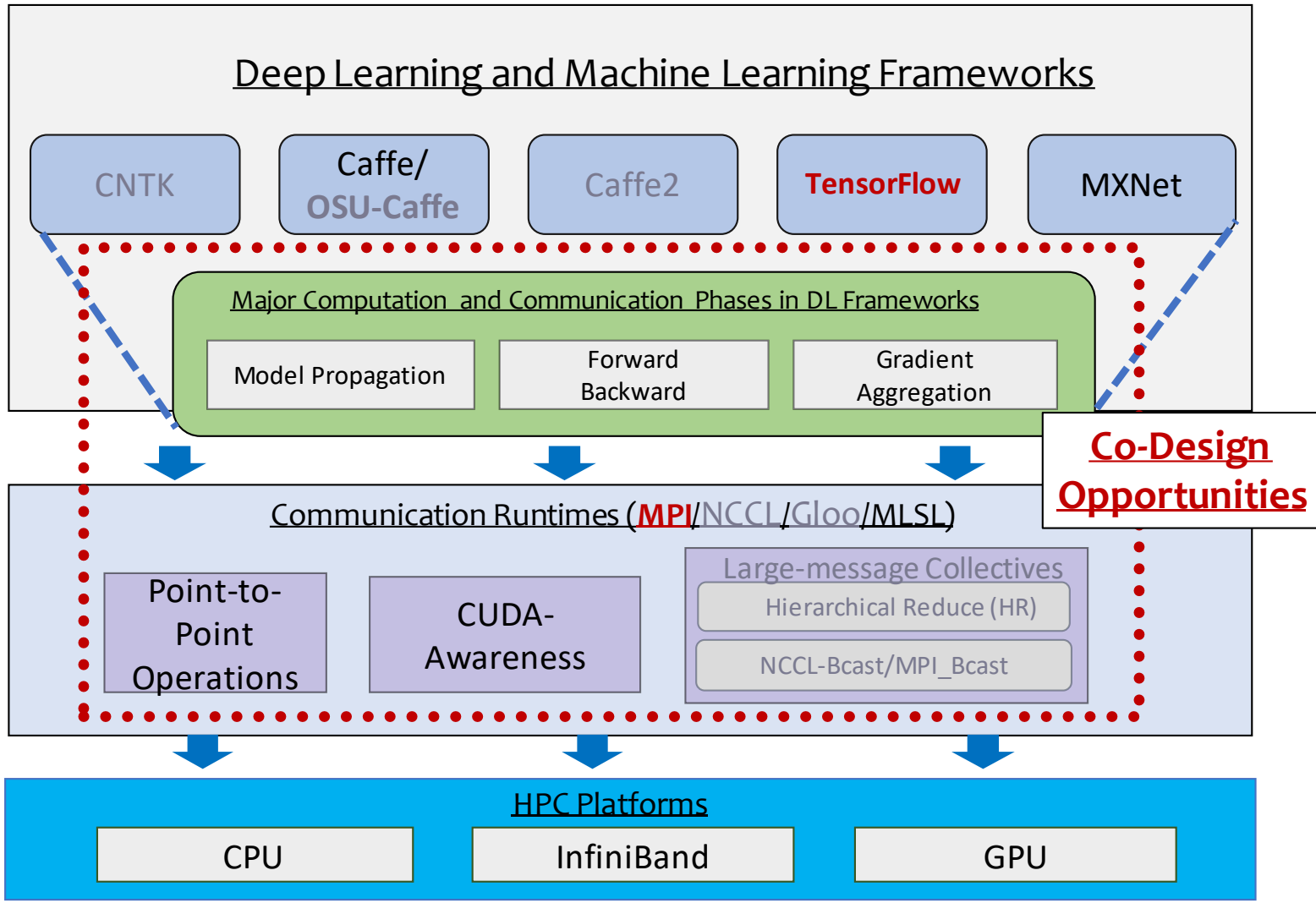
- Caffe : A flexible and layered Deep Learning framework.
- Benefits and Weaknesses
  - Multi-GPU Training within a single node
  - Performance degradation for GPUs across different sockets
  - Limited Scale-out
- OSU-Caffe: MPI-based Parallel Training
  - Enable Scale-up (within a node) and Scale-out (across multi-GPU nodes)
  - Scale-out on 64 GPUs for training CIFAR-10 network on CIFAR-10 dataset
  - Scale-out on 128 GPUs for training GoogLeNet network on ImageNet dataset



A. A. Awan, K. Hamidouche, J. M. Hashmi, and D. K. Panda, S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters. In *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (PPoPP '17)

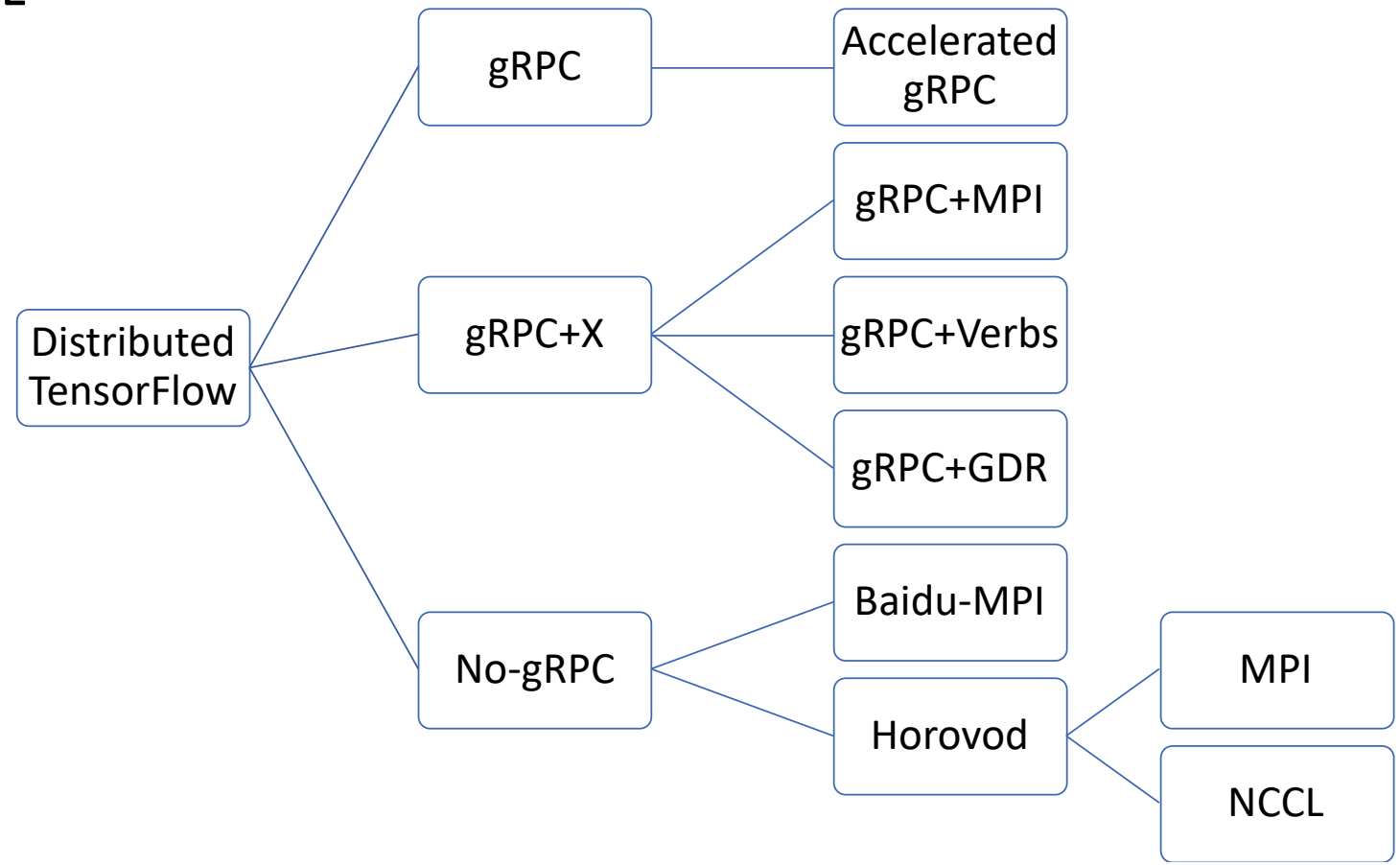
Solutions and Case Studies: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- **Distributed Training for TensorFlow**
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL



Distributed Training using TensorFlow (TF)

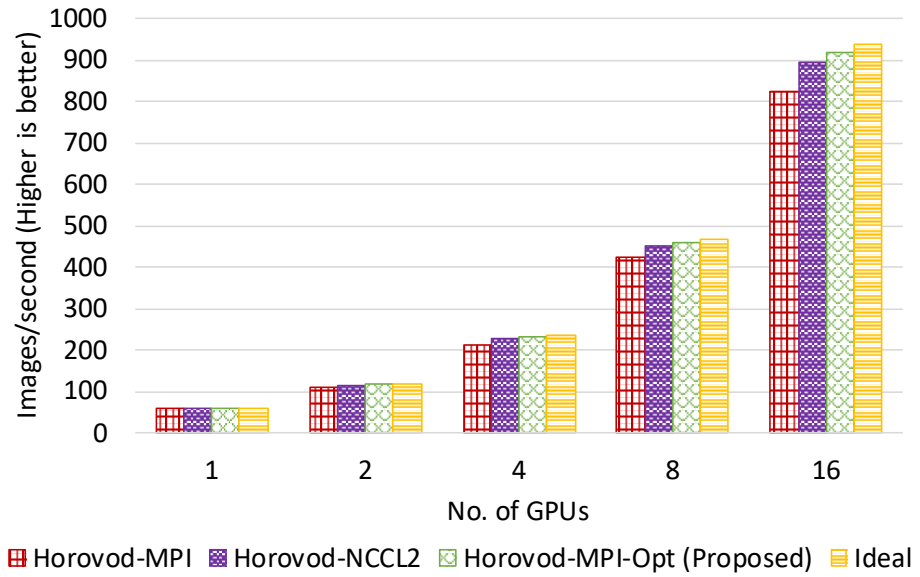
- TensorFlow is the most popular DL framework
- gRPC is the official distributed training runtime
  - Many problems for HPC use-cases
- Community efforts - Baidu and Uber's Horovod have added MPI support to TF across nodes
- Need to understand several options currently available →

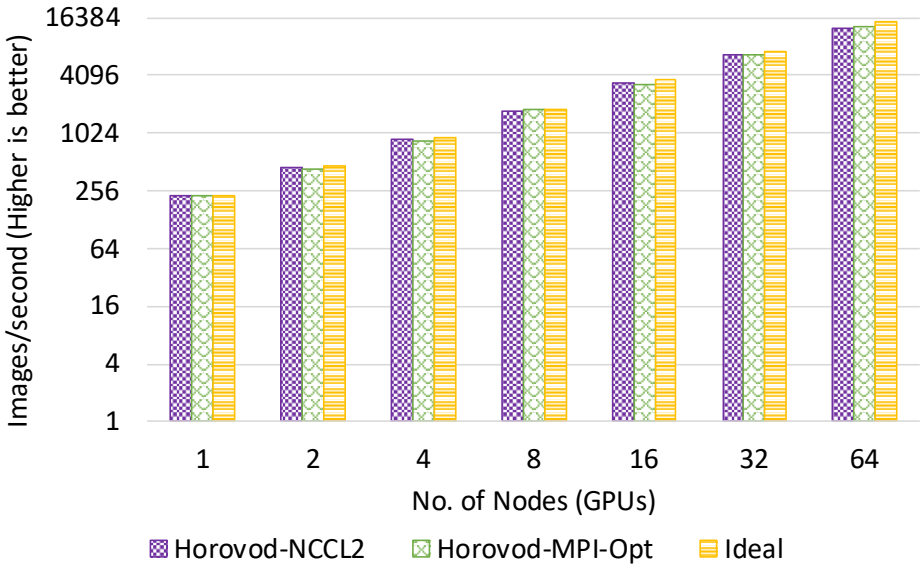


A. A. Awan, J. Bedorf, C.-H. Chu, H. Subramoni and D. K. Panda, "Scalable Distributed DNN Training using TensorFlow and CUDA-Aware MPI: Characterization, Designs, and Performance Evaluation", (To be presented) CCGrid '19. https://arxiv.org/abs/1810.11112

Scalable TensorFlow using Horovod, MPI, and NCCL

- Efficient Allreduce is crucial for Horovod's overall training performance
  - Both MPI and NCCL designs are available
- We have evaluated Horovod extensively and compared across a wide range of designs using gRPC and gRPC extensions
- MVAPICH2-GDR achieved up to ***90%*** scaling efficiency for ResNet-50 Training on 64 Pascal GPUs

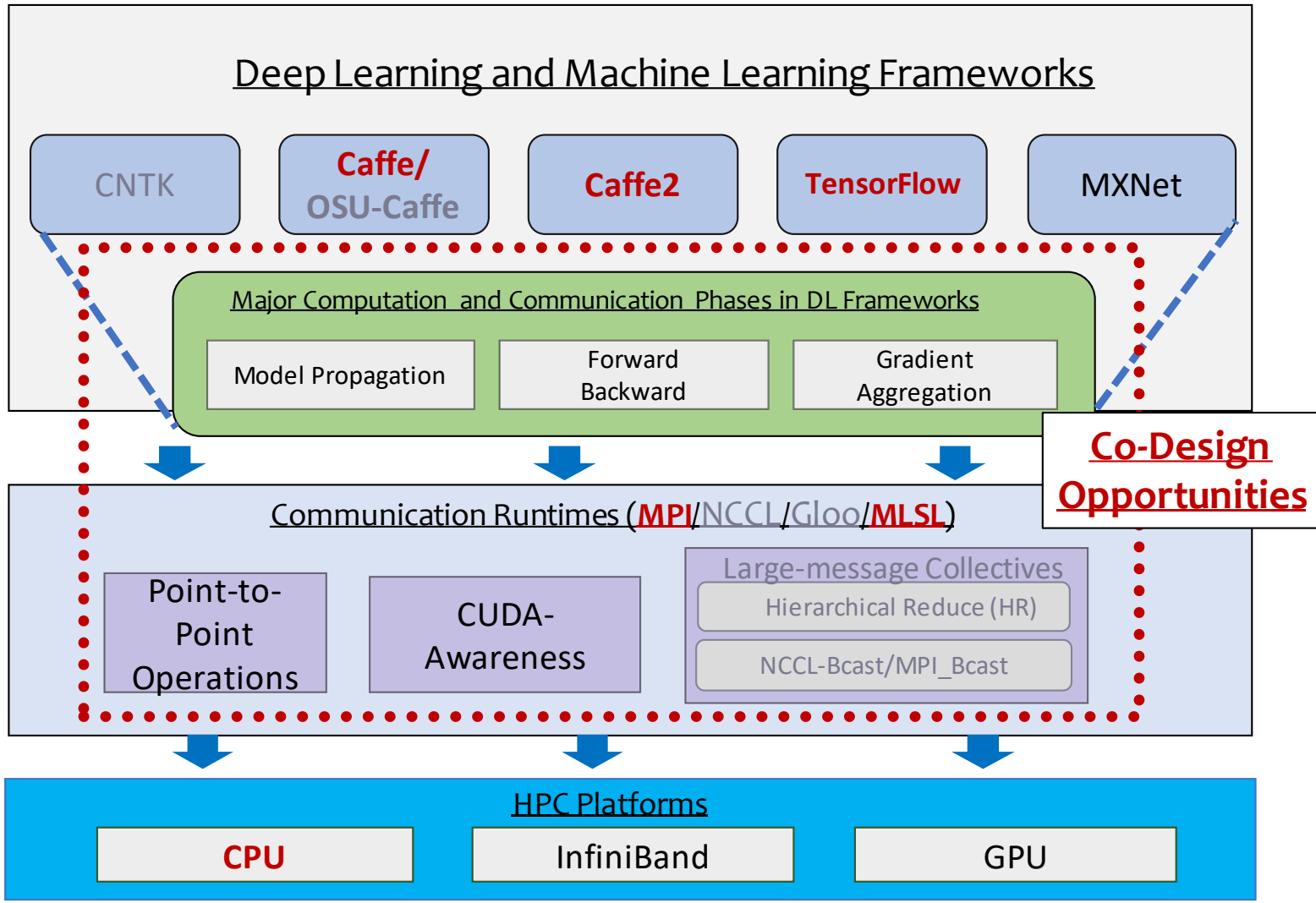




A. A. Awan, J. Bedorf, C.-H. Chu, H. Subramoni and D. K. Panda, "Scalable Distributed DNN Training using TensorFlow and CUDA-Aware MPI: Characterization, Designs, and Performance Evaluation", (To be presented) CCGrid '19.
https://arxiv.org/abs/1810.11112

Solutions and Case Studies: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- **Scaling DNN Training on Multi-/Many-core CPUs**
- PowerAI DDL

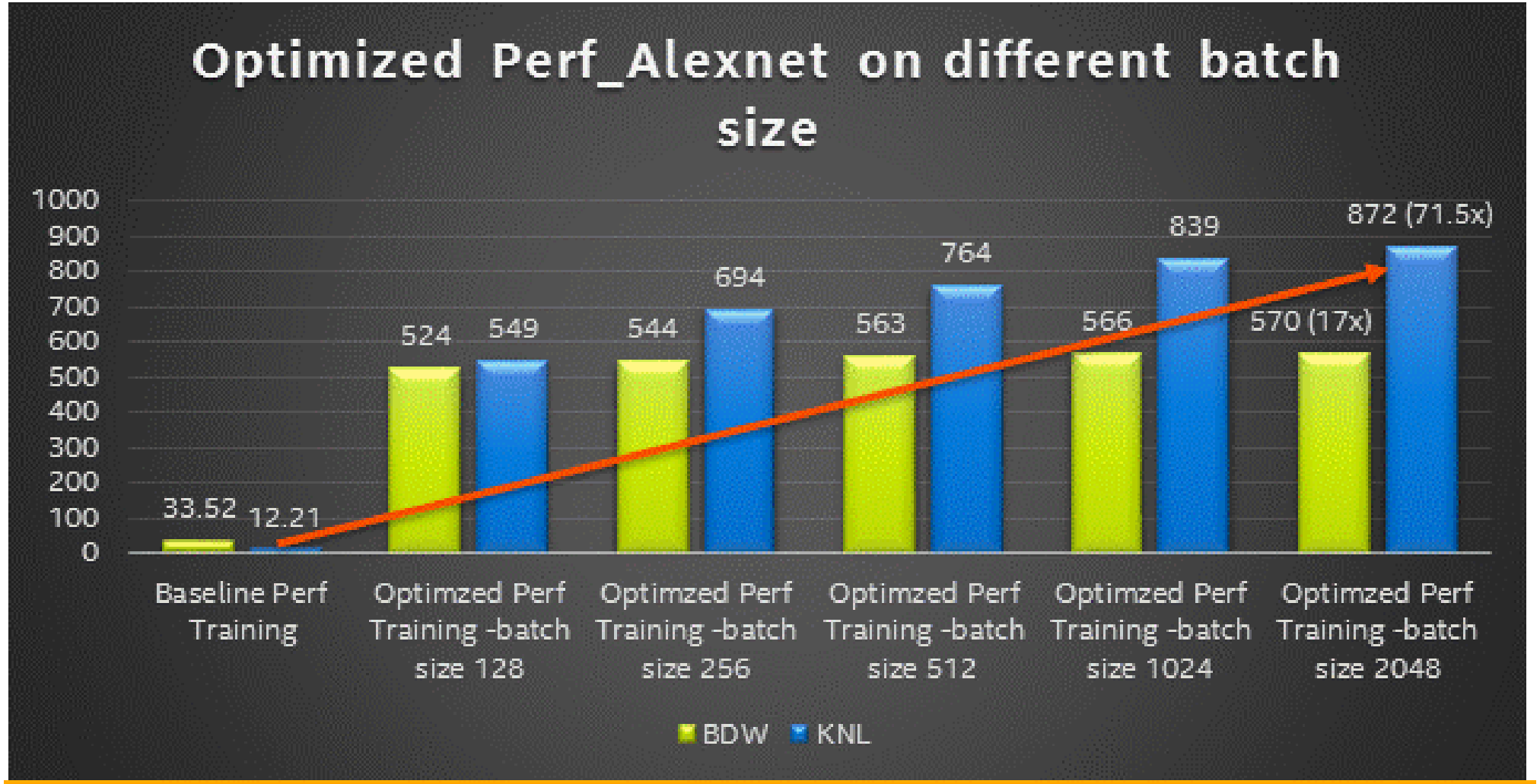


Caffe2 Performance Optimization with Intel MKL

OMP_NUM_THREADS=44		OMP_NUM_THREADS=1		
batch size	Intel® MKL (images/sec)	Eigen BLAS (images/sec)	Intel® MKL (images/sec)	Eigen BLAS (images/sec)
1	173.4	5.2	28.6	5.1
32	1500.2	29.3	64.6	15.4
64	1596.3	35.3	66.0	15.5
256	1735.2	44.9	67.3	16.2

Courtesy: https://software.intel.com/en-us/blogs/2017/04/18/intel-and-facebook-collaborate-to-boost-caffe2-performance-on-intel-cpu-s

TensorFlow Optimization for Intel CPUs



72x Speedup From New Optimizations – available through Google's TensorFlow Git

Courtesy: https://software.intel.com/en-us/articles/tensorflow-optimizations-on-modern-intel-architecture

Intel Machine Learning Scaling Library (MLSL)

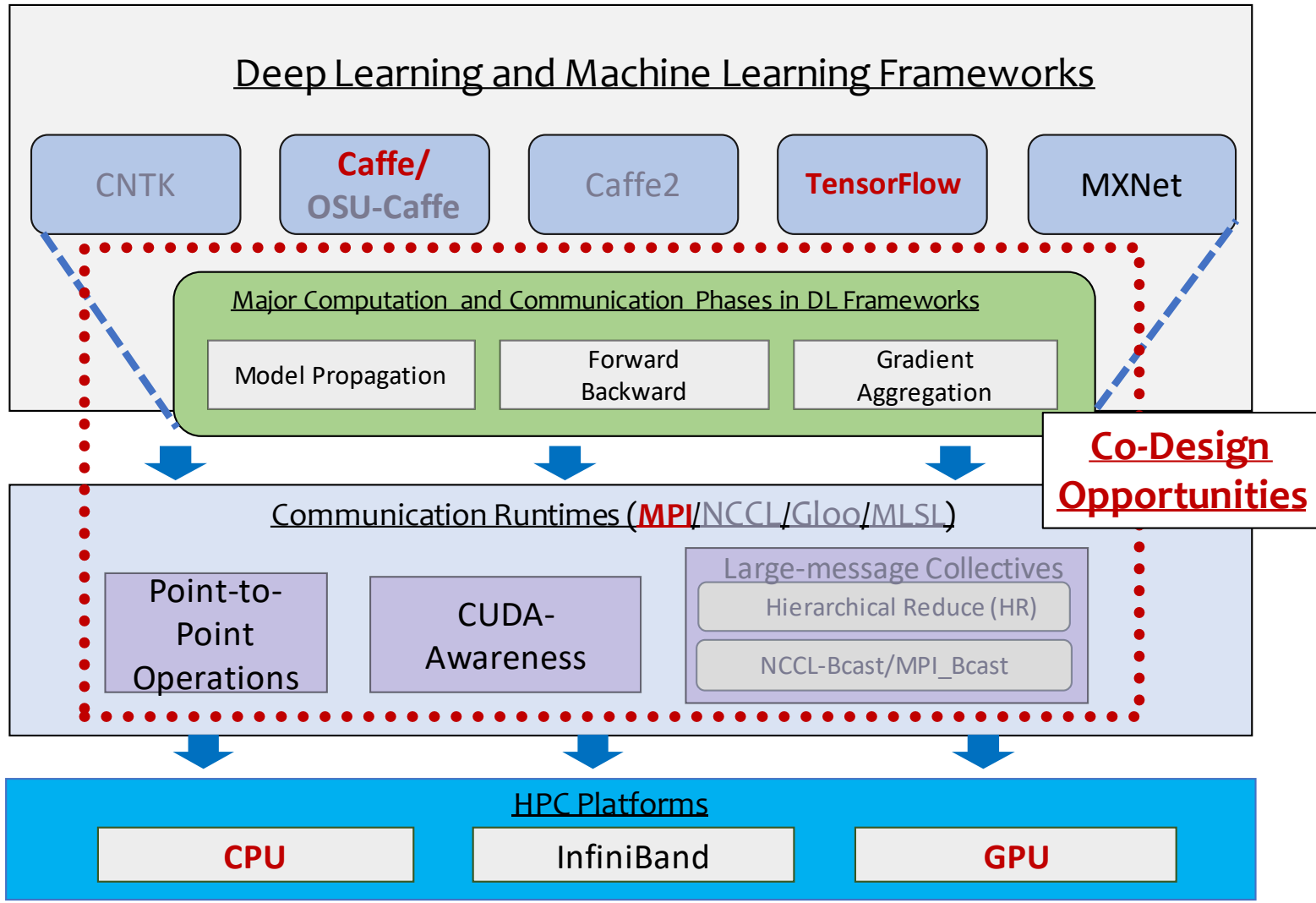
- Intel MLSL is built on top of MPI primitives
  - https://github.com/01org/MLSL
- Works across various interconnects: Intel(R) Omni-Path Architecture, InfiniBand*, and Ethernet
- Common API to support Deep Learning frameworks (Caffe*, Theano*, Torch*, etc.)

MLSL::Activation	A wrapper class for operation input and output activations
MLSL::CommBlockInfo	A class to hold block information for activations packing/unpacking
MLSL::Distribution	A class to hold the information about the parallelism scheme being used
MLSL::Environment	A singleton object that holds global Intel MLSL functions
MLSL::Operation	A class to hold information about learnable parameters (parameter sets) and activations corresponding to a certain operation of the computational graph
MLSL::OperationRegInfo	A class to hold Operation registration information
MLSL::ParameterSet	A wrapper class for operation parameters
MLSL::Session	A class to represent a collection of Operation objects with the same global mini-batch size
MLSL::Statistics	A class to measure and store performance statistics of communication among processes that perform computation in the computational graph

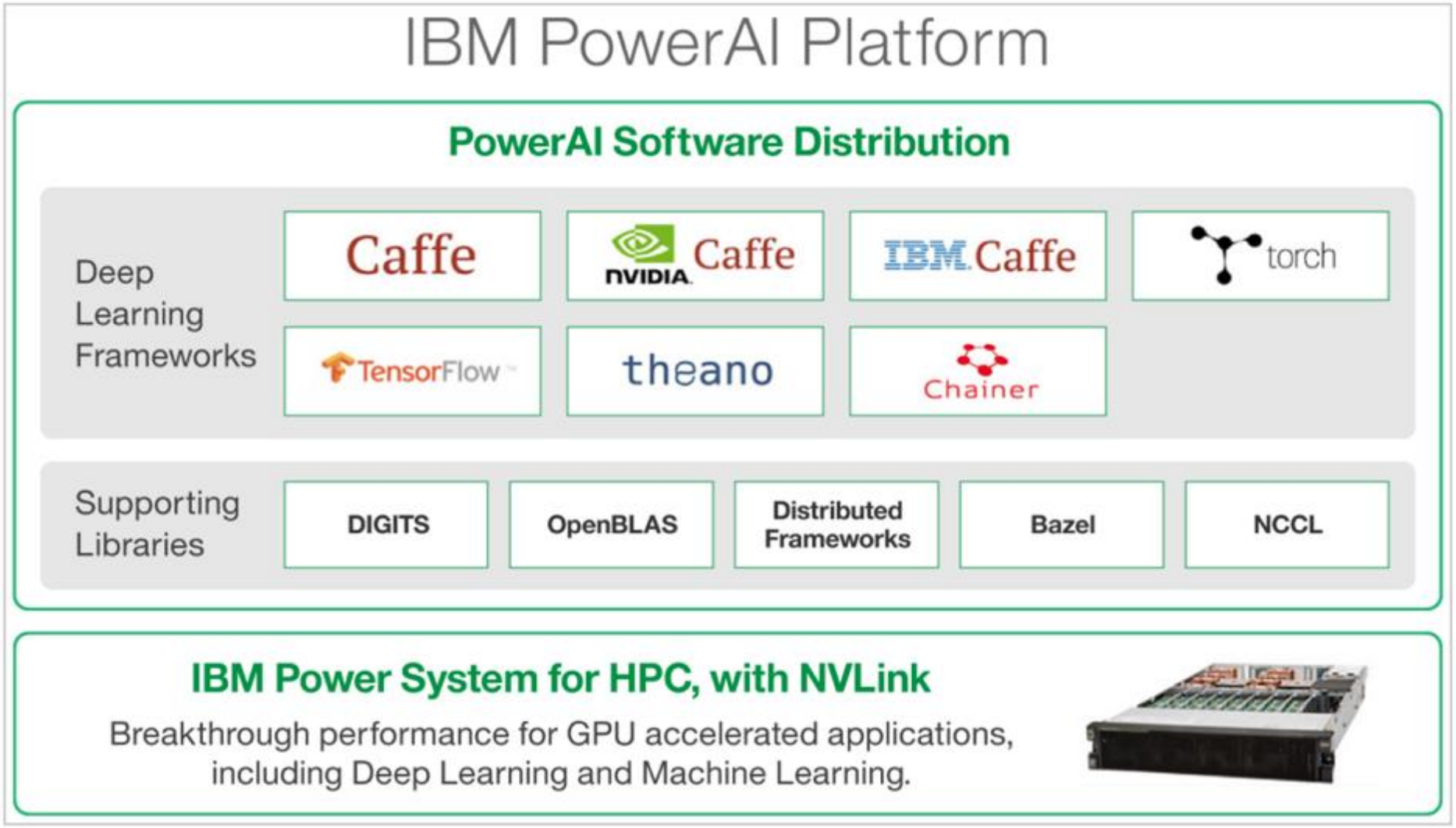
Courtesy: https://github.com/01org/MLSL

Solutions and Case Studies: Exploiting HPC for DL

- NVIDIA NCCL
- LLNL Aluminum
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- **PowerAI DDL**

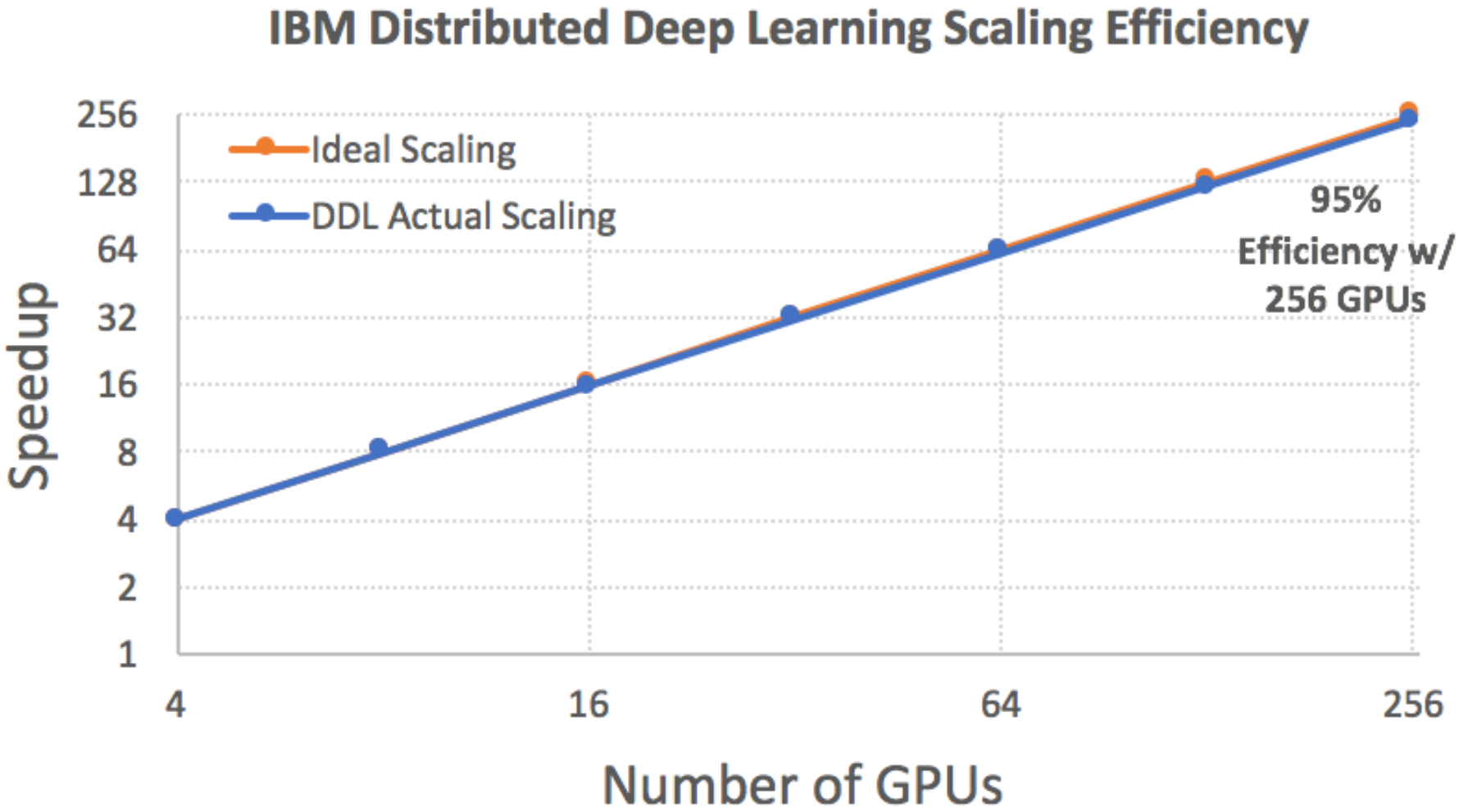


IBM PowerAI DDL



Courtesy: https://www.hpcwire.com/2017/08/08/ibm-raises-bar-distributed-deep-learning/

PowerAI DDL Performance



Caffe with PowerAI DDL on ResNet-50 model using the ImageNet-1K data set on 64 Power8 servers

Courtesy:
https://www.ibm.com/blogs/research/2017/08/distributed-deep-learning/
https://arxiv.org/pdf/1708.02188.pdf

Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- **Open Issues and Challenges**
- Conclusion

Open Issues and Challenges

- Convergence of DL and HPC
- Scalability and Large batch-size training?
- DL Benchmarks and Thoughts on Standardization

Convergence of DL and HPC

- Is Deep Learning an HPC Problem?
  - Distributed DNN Training is definitely an HPC problem
  - Inference – not yet an HPC problem
- Why HPC can help?
  - Decades of research for communication models and performance optimizations
  - MPI, PGAS, and other upcoming programming models and communication runtimes can help for “data-parallel” training
- Some of the needs for DNN training are an exact match
  - Compute intensive problem
- Some needs are new for distributed/parallel communication runtimes
  - Large Message Communication
  - CUDA-Aware Communication

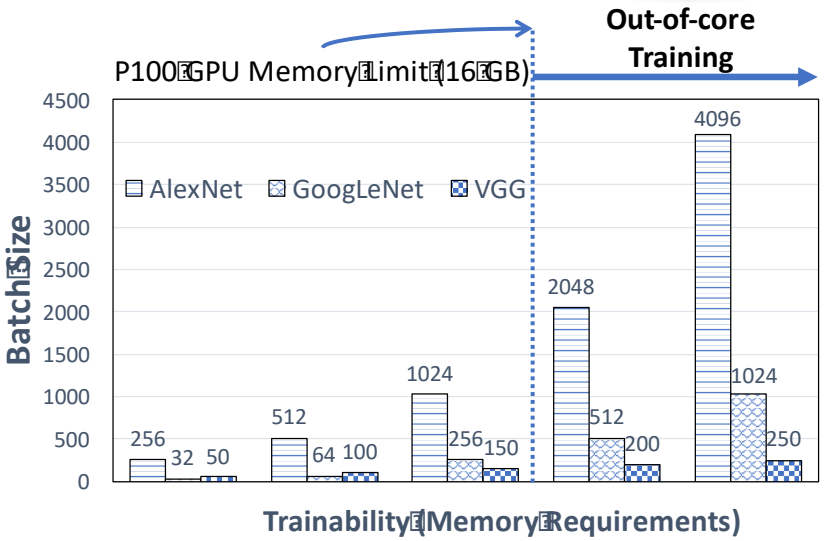
Scalability and Large batch-size training?

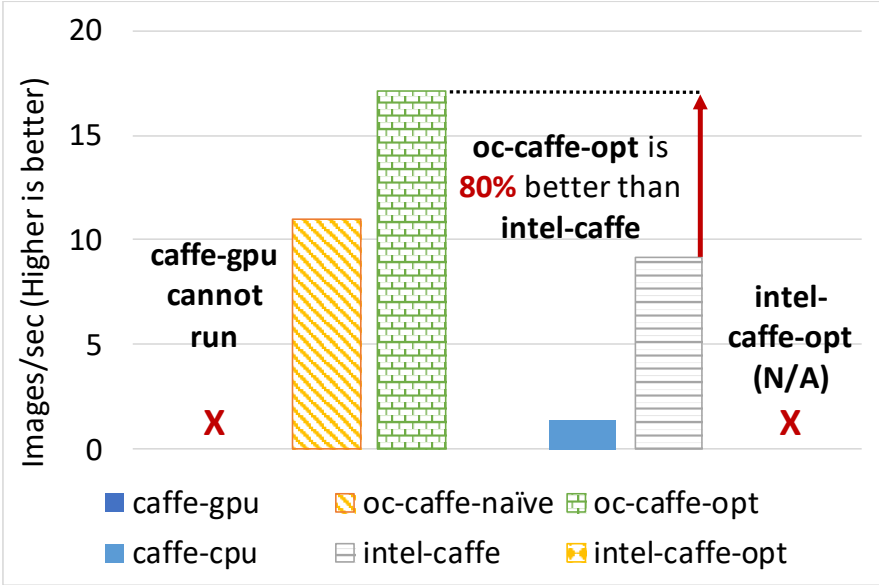
- Large batch-size helps improve the scalability
  - Lesser communication and more compute before synchronization
  - Limits to large batch-size
    - DL community is actively exploring this area
    - HPC community can also investigate overlap and latency-hiding techniques
- Is there a limit to DNN size?
  - Noam Shazeer's Outrageously Large Model (137 Billion Parameters)
  - https://arxiv.org/pdf/1701.06538.pdf
- Out-of-core Training for GPUs?
  - NVIDIA's vDNN - https://arxiv.org/pdf/1602.08124.pdf
  - Prune the network or selectively allocate/de-allocate memory on GPUs
  - OC-DNN and OC-Caffe

Scalability and Large (Out-of-core) Models?

Research Poster Session tonight (P9243)

- Large DNNs cannot be trained on GPUs due to memory limitation!
  - ResNet-50 for Image Recognition but current frameworks can only go up to a small batch size of 45
  - Next generation models like Neural Machine Translation (NMT) are ridiculously large, consists of billions of parameters, and require even more memory
  - Can we design Out-of-core DNN training support using new software features in CUDA 8/9 and hardware mechanisms in Pascal/Volta GPUs?
- General intuition is that managed allocations "will be" slow!
  - The proposed framework called **OC-Caffe (Out-of-Core Caffe)** shows the potential of managed memory designs that can provide performance with negligible/no overhead.
- OC-Caffe-Opt: up to **80% better** than Intel-optimized CPU Caffe for ResNet-50 training on the Volta V100 GPU with CUDA9 and CUDNN7





A. A. Awan, C.-H. Chu, H. Subramoni, X. Lu, and D. K. Panda, OC-DNN: Exploiting Advanced Unified Memory Capabilities in CUDA 9 and Volta GPUs for Out-of-Core DNN Training, HiPC '18

DL Benchmarks and Thoughts on Standardization

- Can we have a standardized interface?
  - Are we there yet?
  - Deep Learning Interface (DLI)? Inspired by Message Passing Interface (MPI)
    - What can be a good starting point?
    - Will it come from the HPC community or the DL community?
    - Can there be a collaboration across communities?
- What about standard benchmarks? Is there a need?
  - State-of-the-art
    - HKBU benchmarks - http://dlbench.comp.hkbu.edu.hk
    - Soumith Chintala's benchmarks - https://github.com/soumith/convnet-benchmarks
    - DAWN Bench – https://dawn.cs.stanford.edu/benchmark/
    - MLPerf – https://www.mlperf.org -- Latest and Widely Promoted now!

Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- **Conclusion**

Conclusion

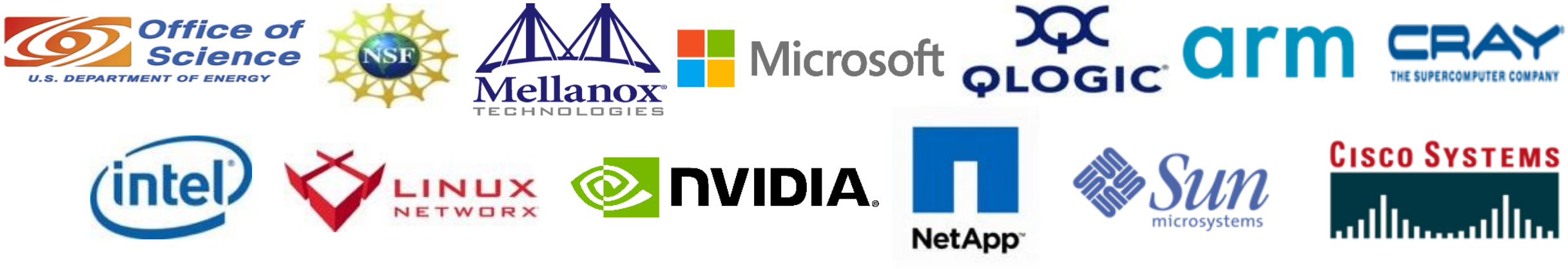
- Exponential growth in Deep Learning frameworks
- Provided an overview of issues, challenges, and opportunities for communication runtimes
  - Efficient, scalable, and hierarchical designs are crucial for DL frameworks
  - Co-design of communication runtimes and DL frameworks will be essential
    - OSU-Caffe
    - TensorFlow (Baidu, Uber's Horovod, etc.)
    - Neon and Nervana Graph
- Need collaborative efforts to achieve the full potential
- Standardization may help remove fragmentation in DL frameworks

Please join us for more events..

Monday, March 18	Tuesday, March 19	Wednesday, March 20
Research Poster 1. **P9243 -** Exploiting CUDA Unified Memory for Efficient Out-of-Core DNN Training 2. **P9242** - Exploiting GPUDirect Technology and Hardware Multicast for Streaming and Deep Learning Applications	**Talk** **S9476** - MVAPICH2-GDR: High-Performance and Scalable CUDA-Aware MPI Library for HPC and AI	**Instructor-Led Training** **L9121** - How to Boost the Performance of HPC/AI Applications Using MVAPICH2 Library
SJCC Upper Concourse 06:00 PM - 08:00 PM	SJCC Room 211A (Concourse Level) 03:00 PM - 03:50 PM	SJCC Room LL21D (Lower Level) 08:00 AM - 10:00 AM

Funding Acknowledgments

Funding Support by



Equipment Support by

Personnel Acknowledgments

Current Students (Graduate)

- A. Awan (Ph.D.)
- M. Bayatpour (Ph.D.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- S. Guganani (Ph.D.)
- J. Hashmi (Ph.D.)
- A. Jain (Ph.D.)
- K. S. Khorassani (Ph.D.)
- P. Kousha (Ph.D.)
- D. Shankar (Ph.D.)

Current Students (Undergraduate)

- V. Gangal (B.S.)
- M. Haupt (B.S.)
- N. Sarkauskas (B.S.)
- A. Yeretzian (B.S.)

Current Research Asst. Professor

- X. Lu

Current Research Scientist

- H. Subramoni

Current Post-doc

- A. Ruhela
- K. Manian

Current Research Specialist

- J. Smith

Past Students

- A. Augustine (M.S.)
- P. Balaji (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.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- 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.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (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 Post-Docs

- D. Banerjee
- X. Besseron
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne
- H. Wang

Past Research Scientist

- K. Hamidouche
- S. Sur

Past Programmers

- D. Bureddy
- J. Perkins

Past Research Specialist

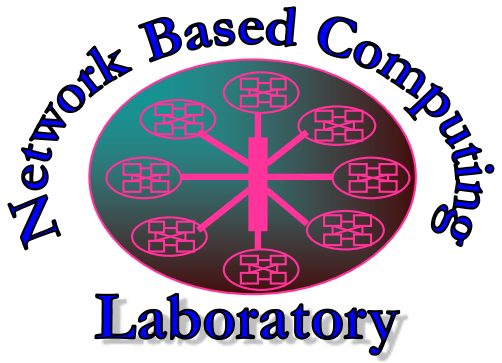
- M. Arnold

Thank You!

panda@cse.ohio-state.edu

awan.10@osu.edu

subramon@cse.ohio-state.edu



Network-Based Computing Laboratory

http://nowlab.cse.ohio-state.edu/

The MVAPICH2 Project

http://mvapich.cse.ohio-state.edu/

The High-Performance Deep Learning Project

http://hidl.cse.ohio-state.edu/