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/hoti17-dl.pdf

High Performance Distributed Deep Learning for Dummies

A Tutorial at Hot Interconnects '17

by

Dhabaleswar K. (DK) Panda
The Ohio State University
E-mail: panda@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~panda

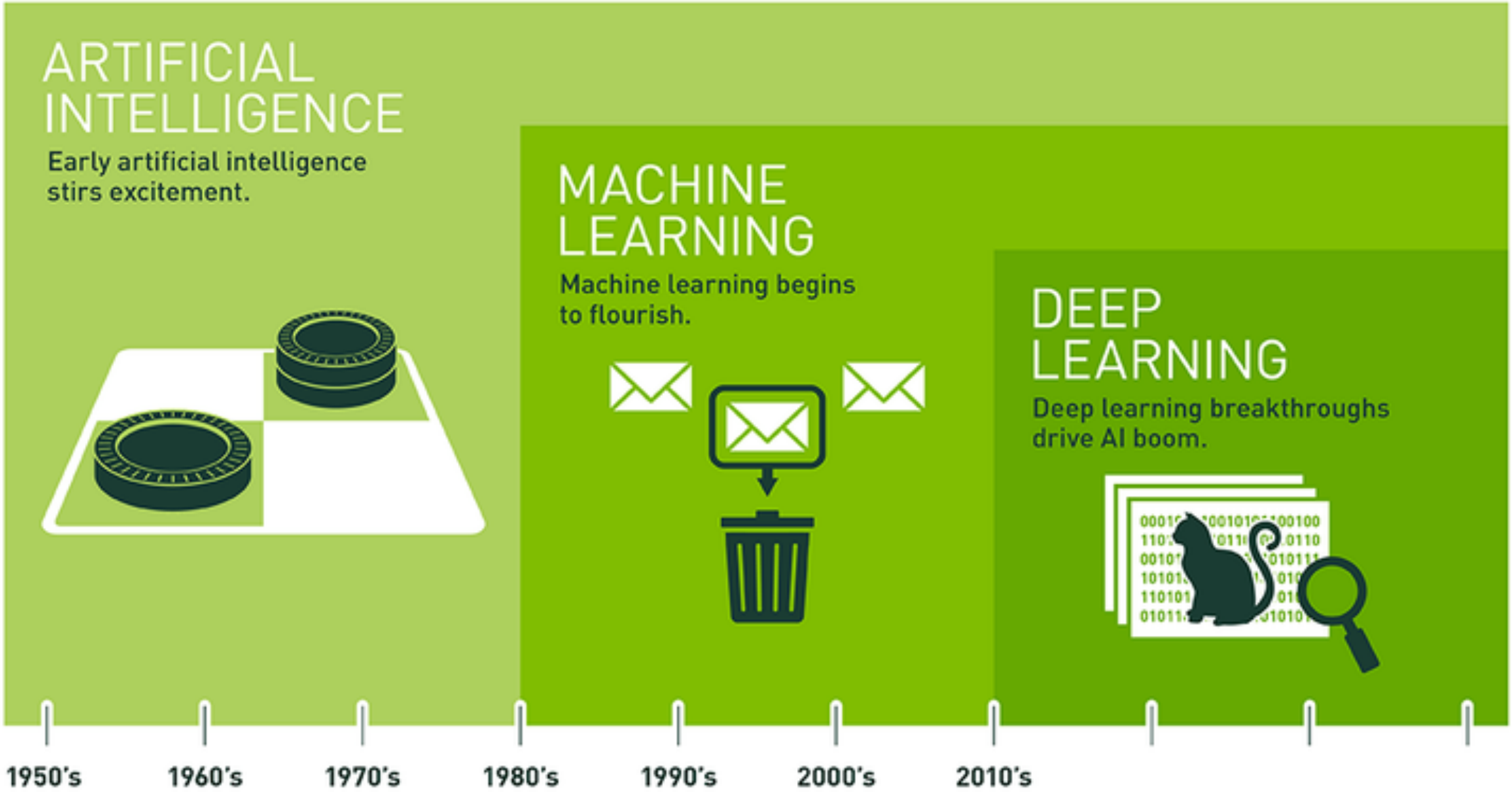
Ammar Ahmad Awan
The Ohio State University
E-mail: awan.10@osu.edu
http://www.cse.ohio-state.edu/~awan.10

Hari Subramoni
The Ohio State University
E-mail: subramon@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~subramon

Outline

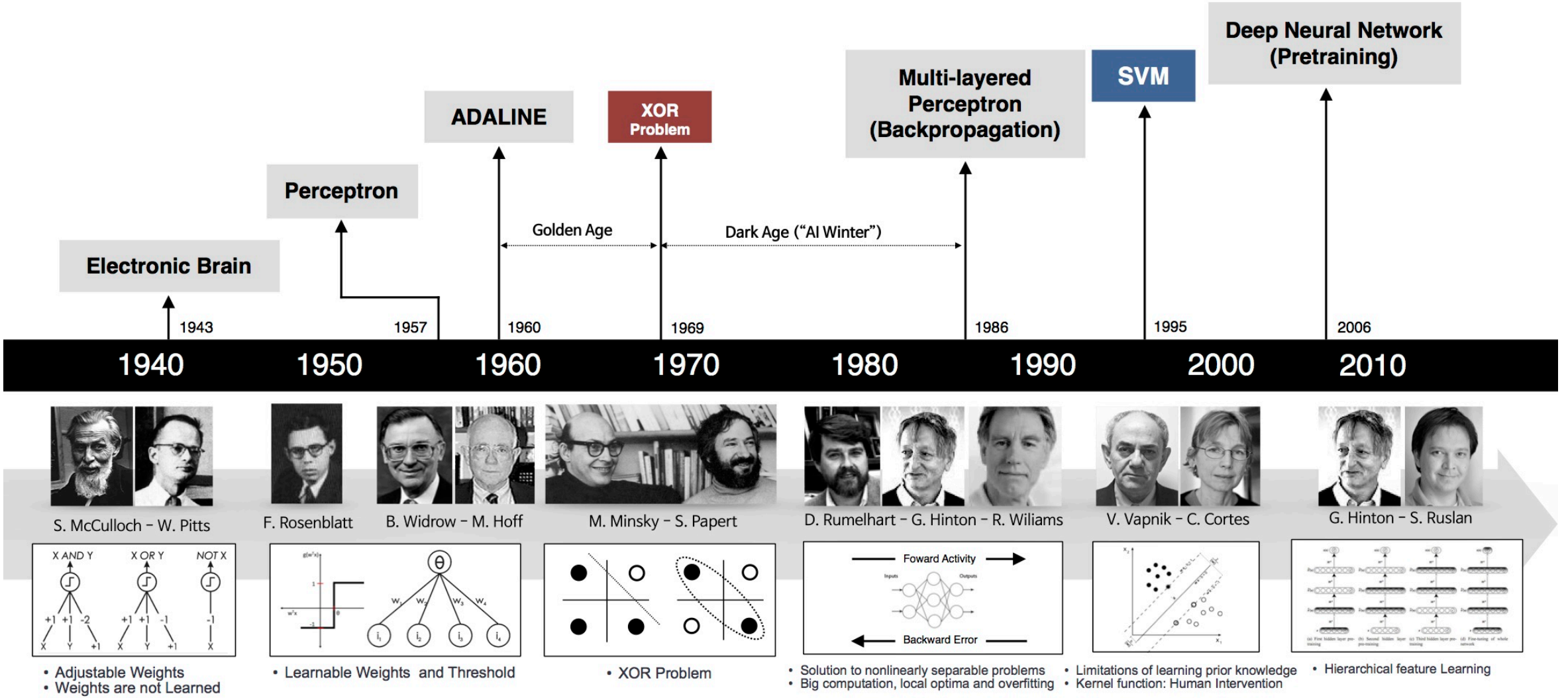
- **Introduction**
  - **The Past, Present, and Future of Deep Learning**
  - What are Deep Neural Networks?
  - Diverse Applications of Deep Learning
  - Deep Learning Frameworks
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges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

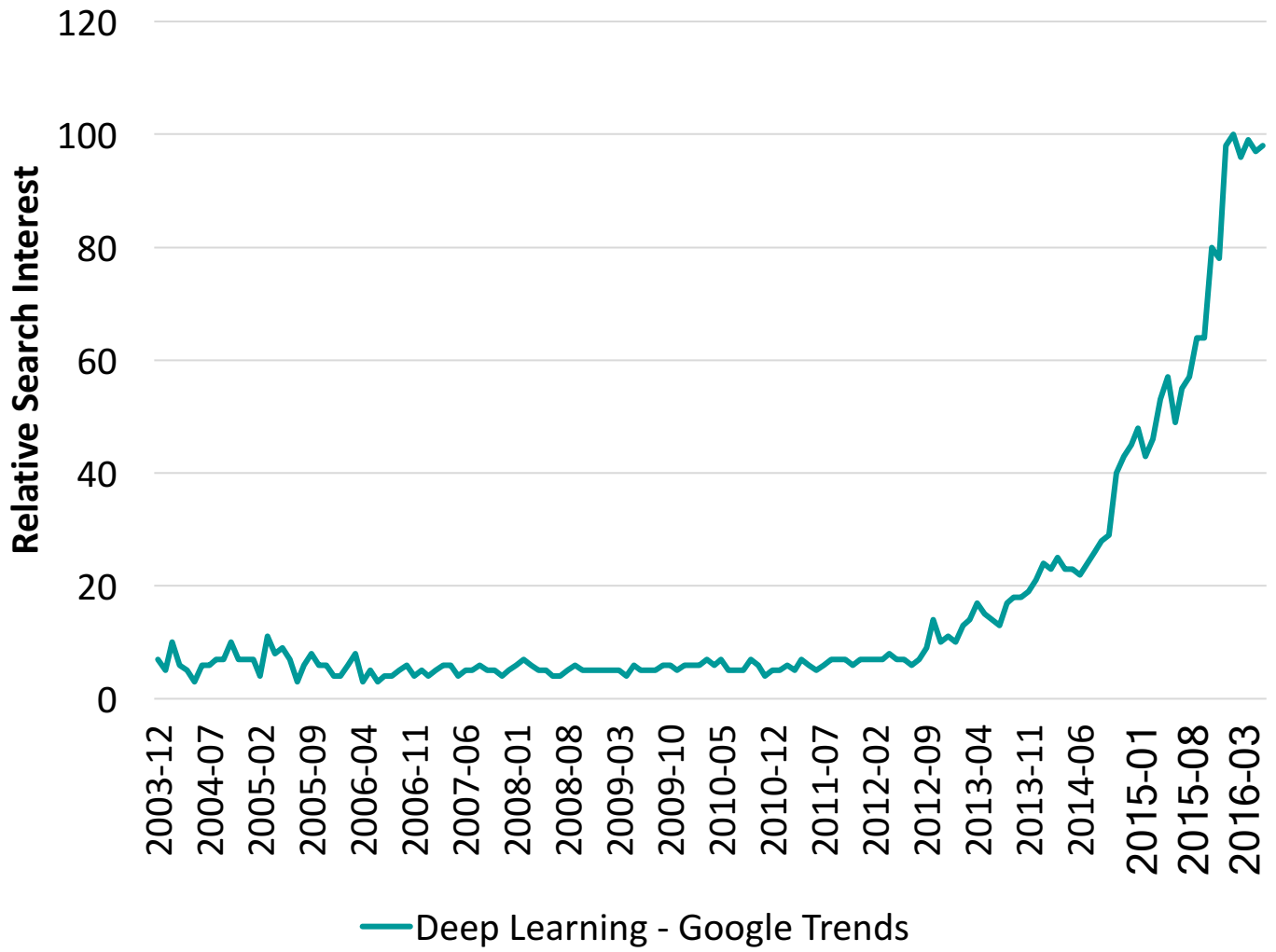
Milestones in the Development of Neural Networks



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

Deep Learning Resurgence

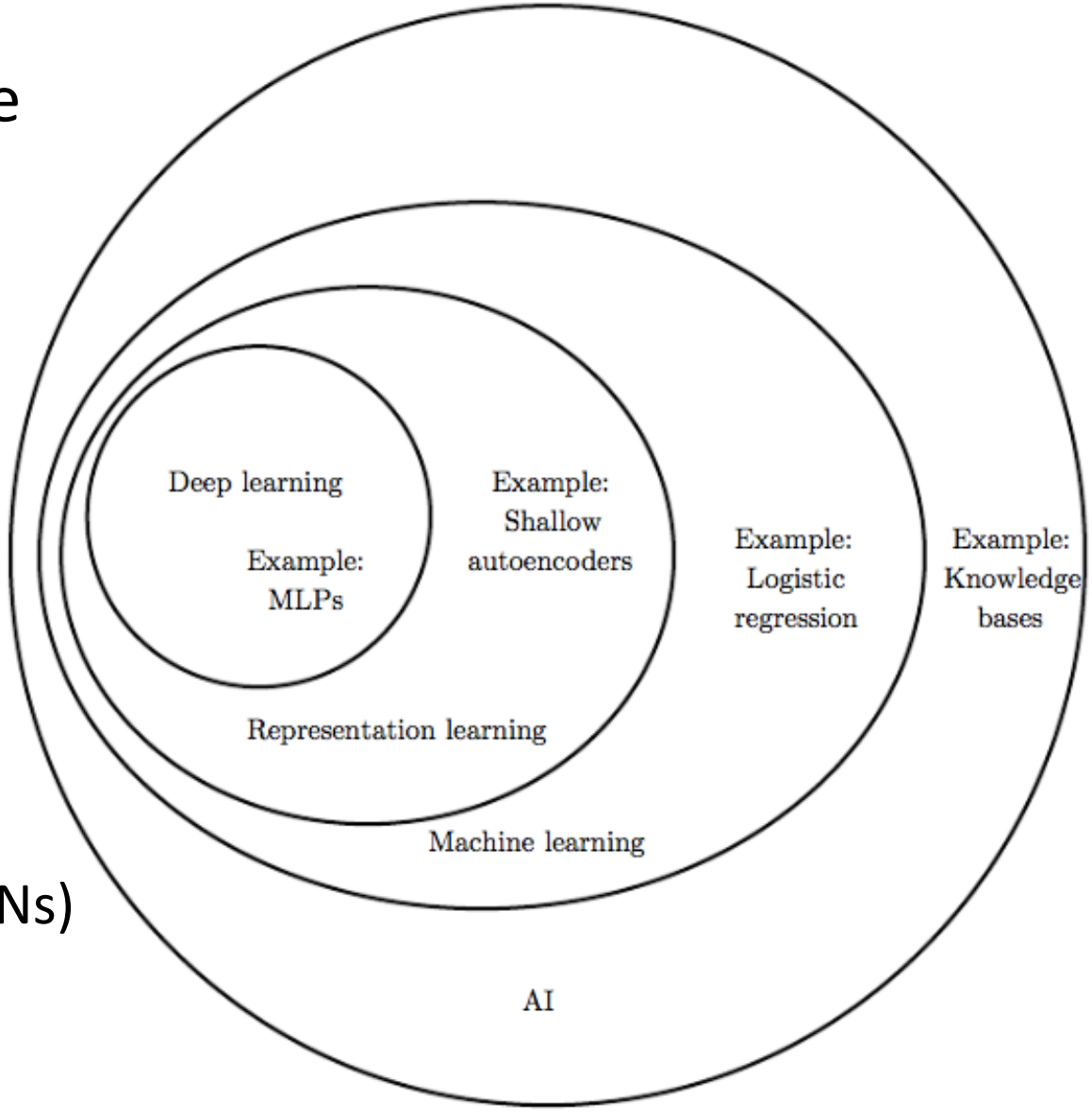
- Deep Learning is going through a resurgence
- Excellent accuracy for deep/convolutional neural networks
- Public availability of versatile datasets like MNIST, CIFAR, and ImageNet
- Widespread popularity of accelerators like NVIDIA GPUs



Courtesy: http://trends.google.com

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**"

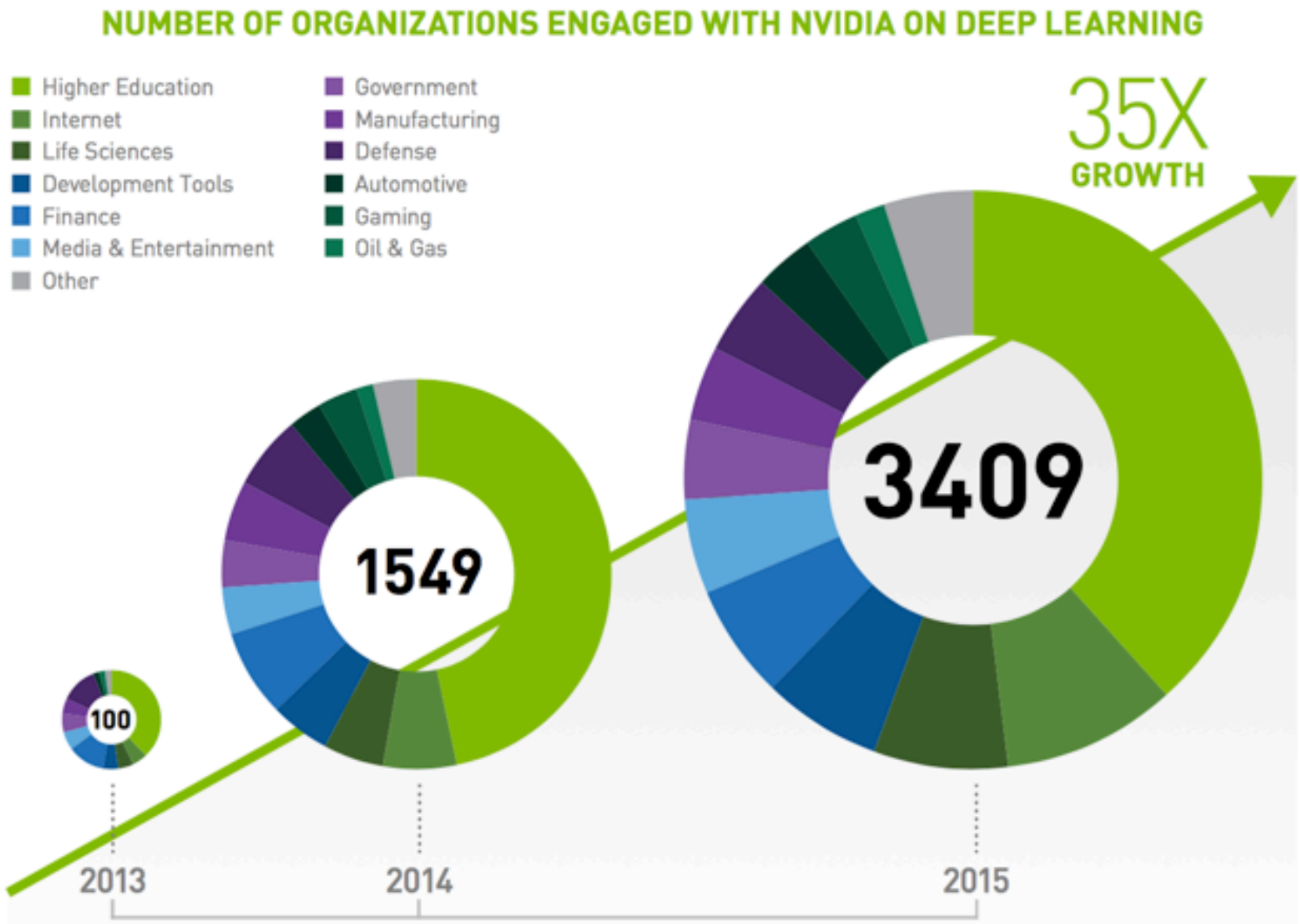


Courtesy: http://www.deeplearningbook.org/contents/intro.html

Resurgence of Deep Learning in the Many-core Era

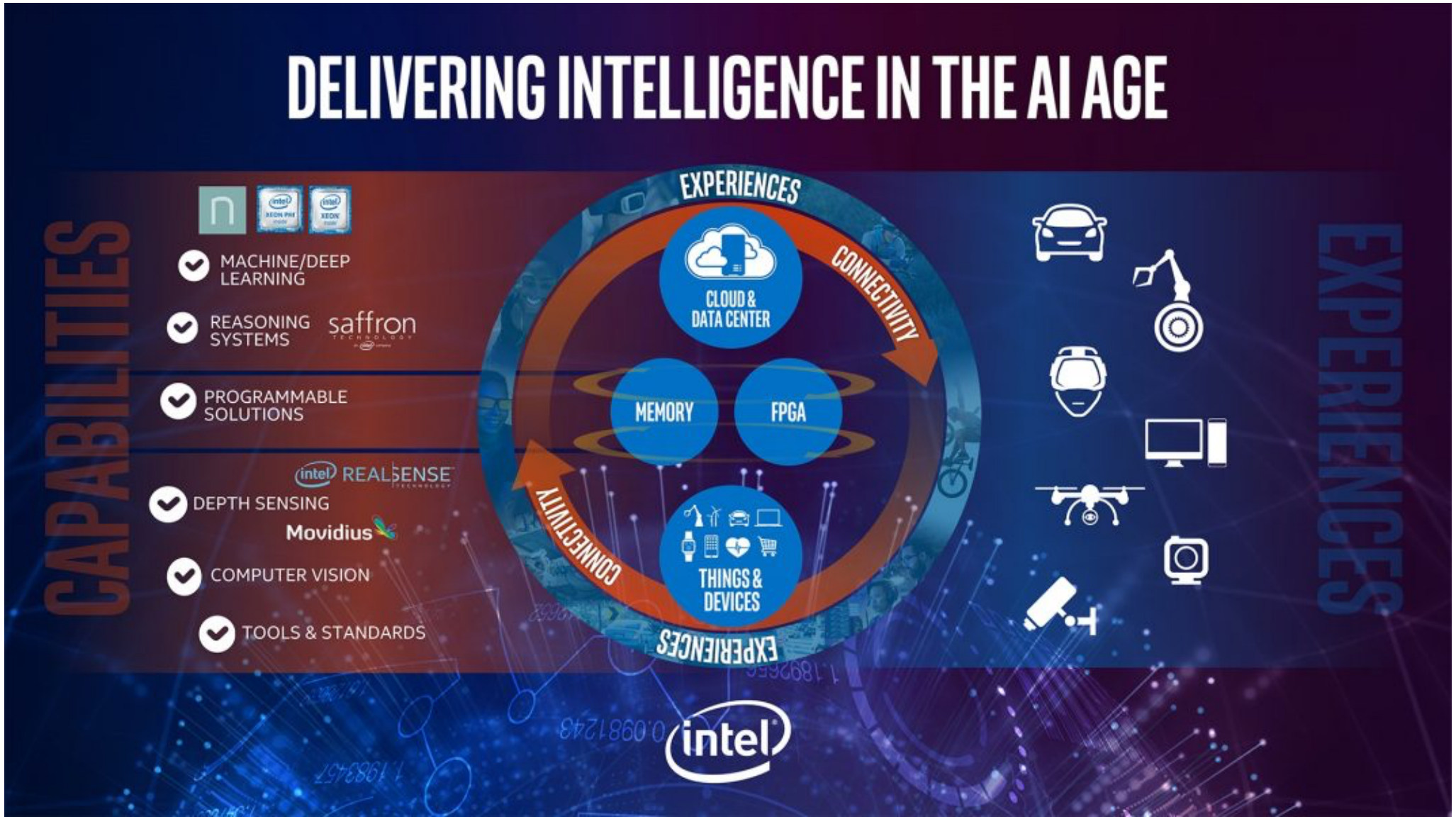
- *Computability of DNNs* made possible by modern and efficient hardware
  - Many DNN training tasks were impossible to compute!
  - GPUs are at the core of DNN training performance!
- Availability of Datasets
  - MNIST - http://yann.lecun.com/exdb/mnist/
  - CIFAR10 - https://www.cs.toronto.edu/~kriz/cifar.html
  - ImageNet - https://www.image-net.org
  - Street View House Numbers (SVHN) - http://ufldl.stanford.edu/housenumbers/
  - Several others..
- Excellent Accuracy for classical Machine Learning problems
  - Case study: 30 years of research vs. proposed Neural Machine Translation
  - https://arxiv.org/abs/1703.01619

The Rise of GPU-based Deep Learning



Courtesy: http://images.nvidia.com/content/technologies/deep-learning/pdf/NVIDIA-DeepLearning-Infographic-v11.pdf

Intel is committed to AI and Deep Learning as well!

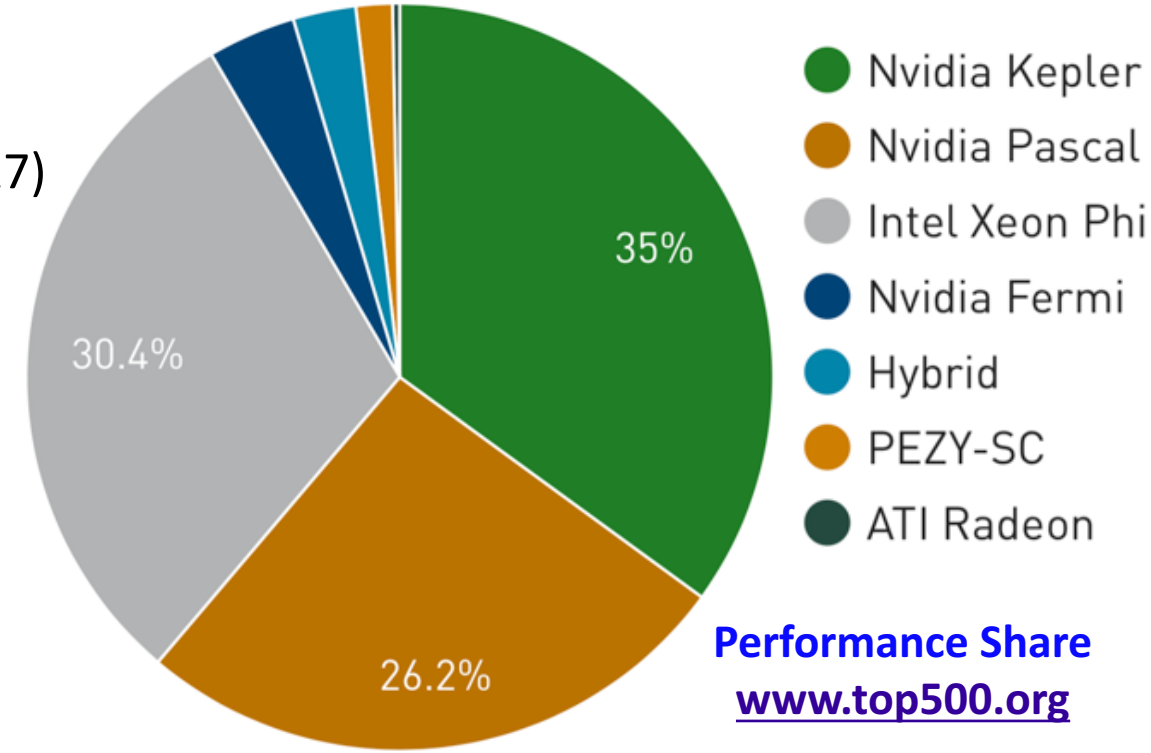


Courtesy: https://newsroom.intel.com/editorials/krzanich-ai-day/

Deep Learning, GPUs, 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
  - 71/500 Top HPC systems use NVIDIA GPUs (Jun '17)
  - CUDA-Aware Message Passing Interface (MPI)
  - Nvidia Fermi, Kepler, and Pascal architecture
  - DGX-1 and DGX1-V (Volta architecture)
    - Dedicated DL super-computers

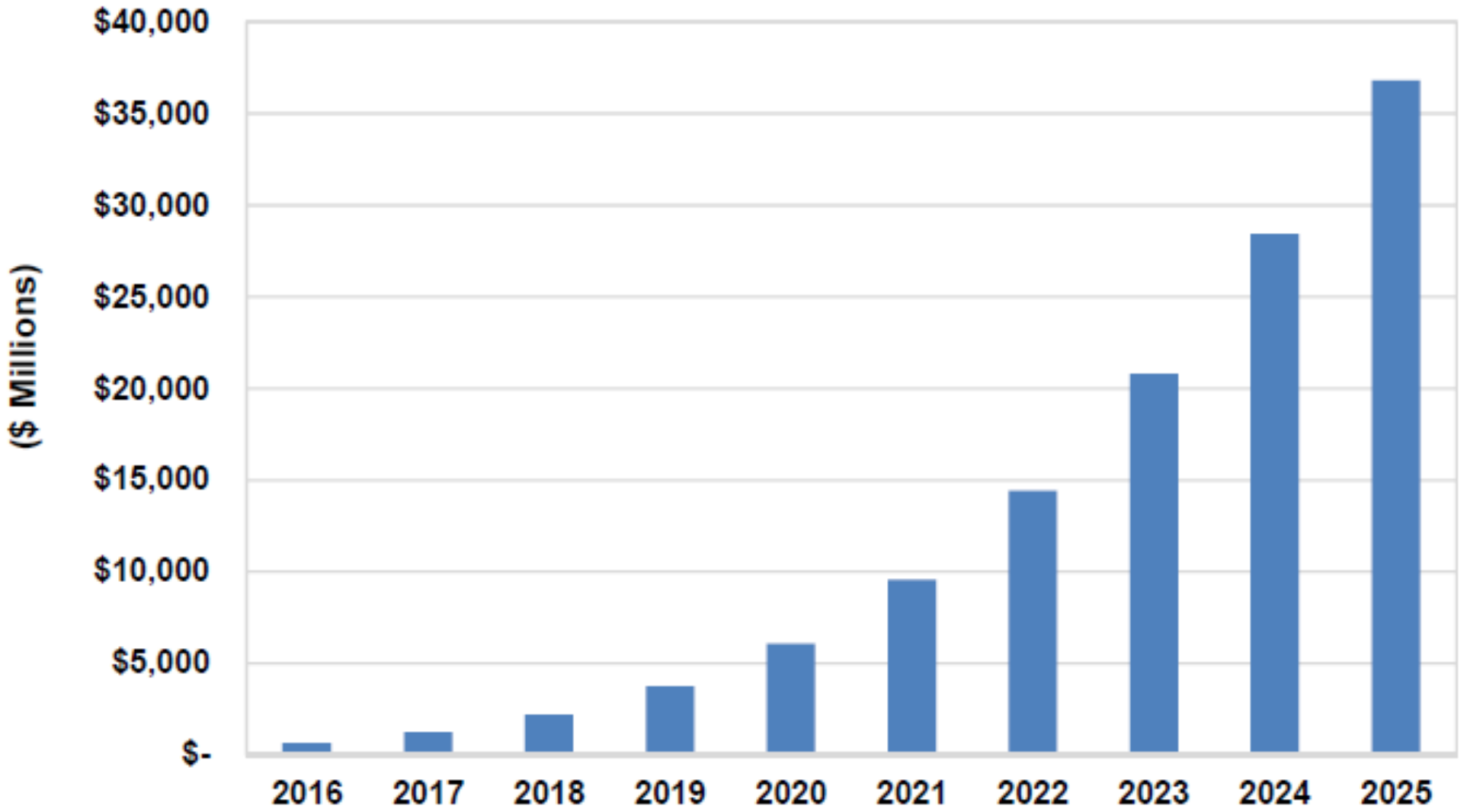


Performance Share
www.top500.org

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

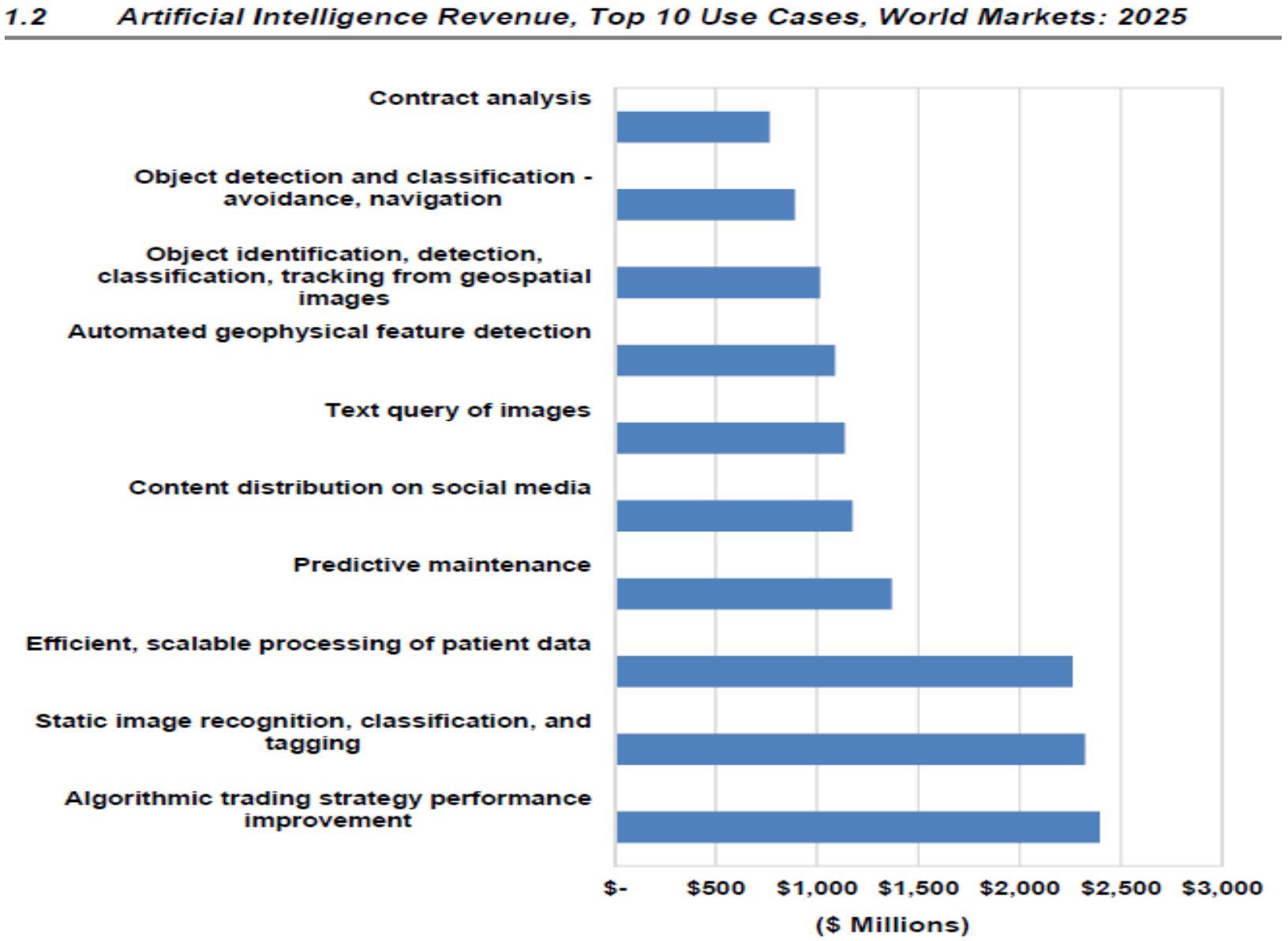
The Bright Future of Deep Learning

1.1 Artificial Intelligence Revenue, World Markets: 2016-2025



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

Current and Future Use Cases of Deep Learning



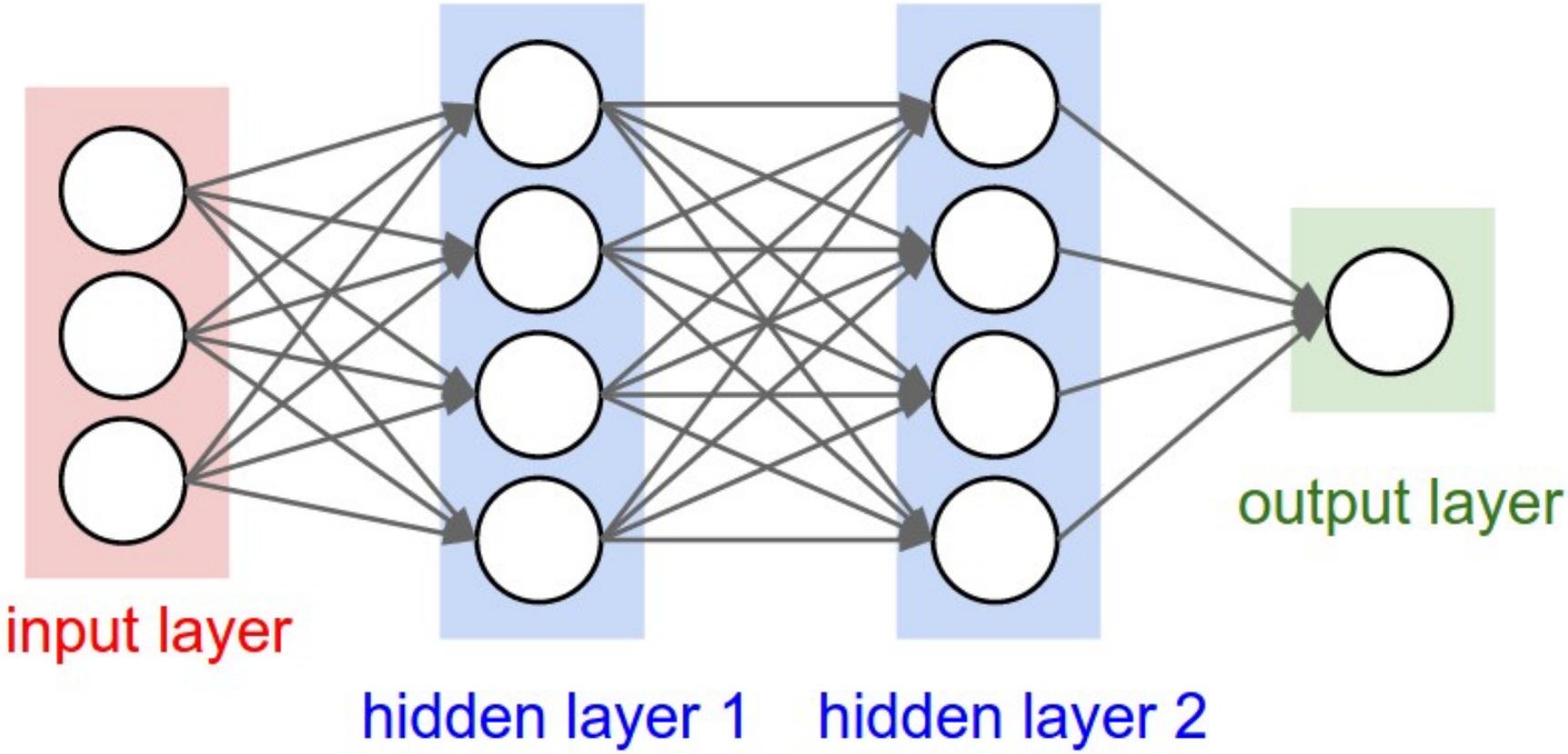
Courtesy: https://www.top500.org/news/market-for-artificial-intelligence-projected-to-hit-36-billion-by-2025/

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
- Case Studies and Demos
- Open Issues and Challenges
- Conclusion

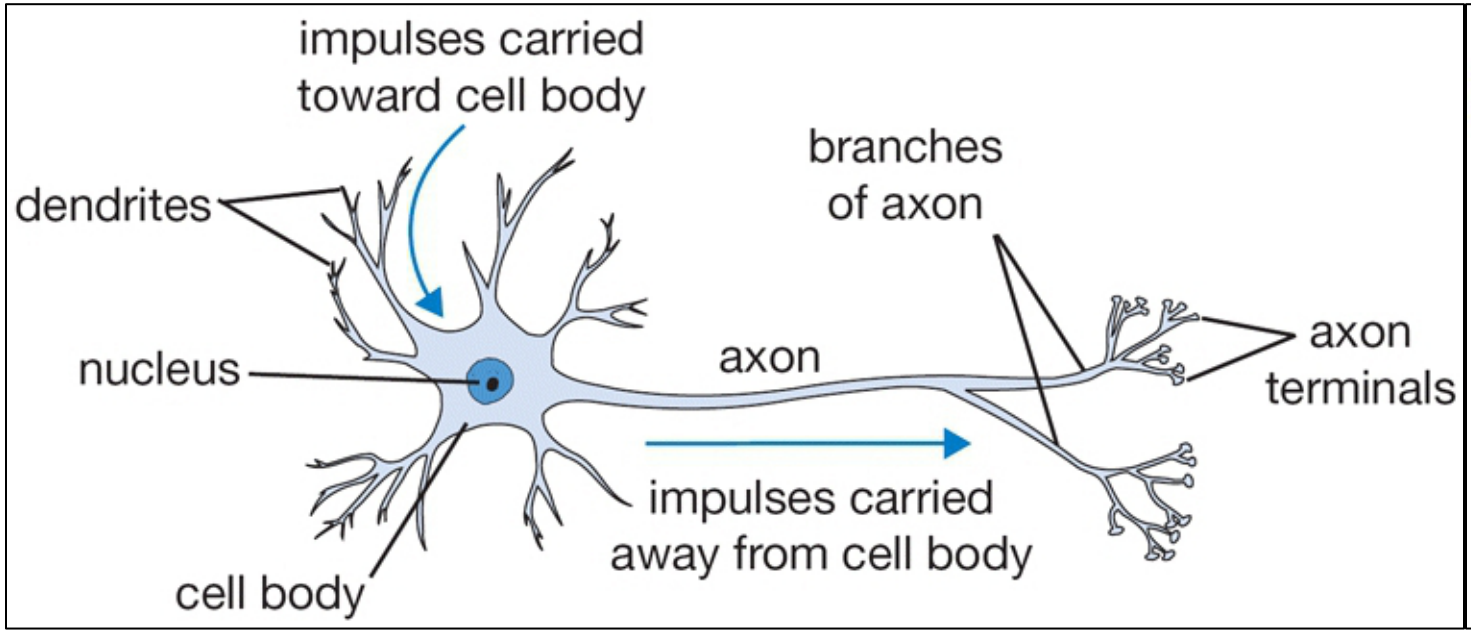
So what is a Deep Neural Network?

- Example of a 2-layer Deep Neural Network (DNN)

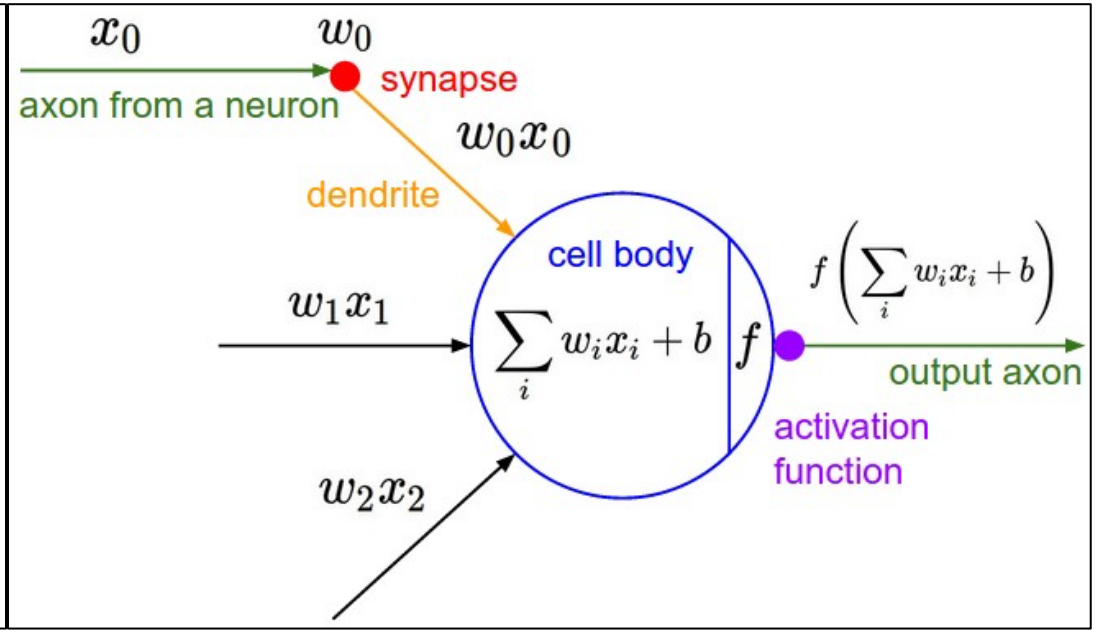


Courtesy: http://cs231n.github.io/neural-networks-1/

Graphical/Mathematical Intuitions for DNNs



Drawing of a Biological Neuron



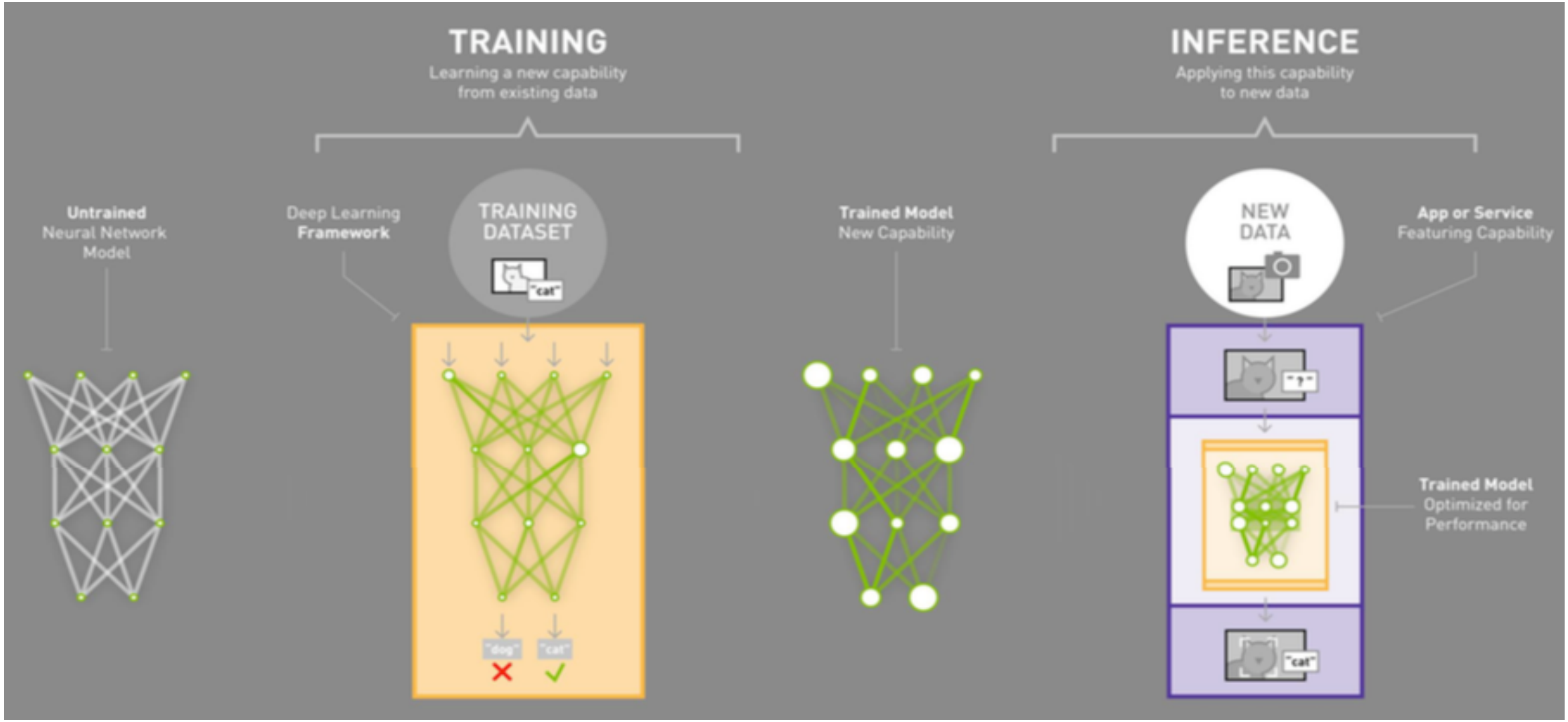
The Mathematical Model

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

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
- Case Studies and Demos
- Open Issues and Challenges
- Conclusion

Diverse Application Areas for Deep Learning

- Vision
  - Image Classification
  - Style Transfer
  - Caption Generation
- Speech
  - Speech Recognition
  - Real-time Translation
- Text
  - Sequence Recognition and Generation
- Disease discovery
  - Cancer Detection
- Autonomous Driving
  - Combination of multiple areas like Image/Object Detection, Speech Recognition, etc.

Style Transfer

Synthesized Image

#NeuralDoodle

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

Style Transfer

Synthesized Image

#NeuralDoodle

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

Caption Generation

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

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

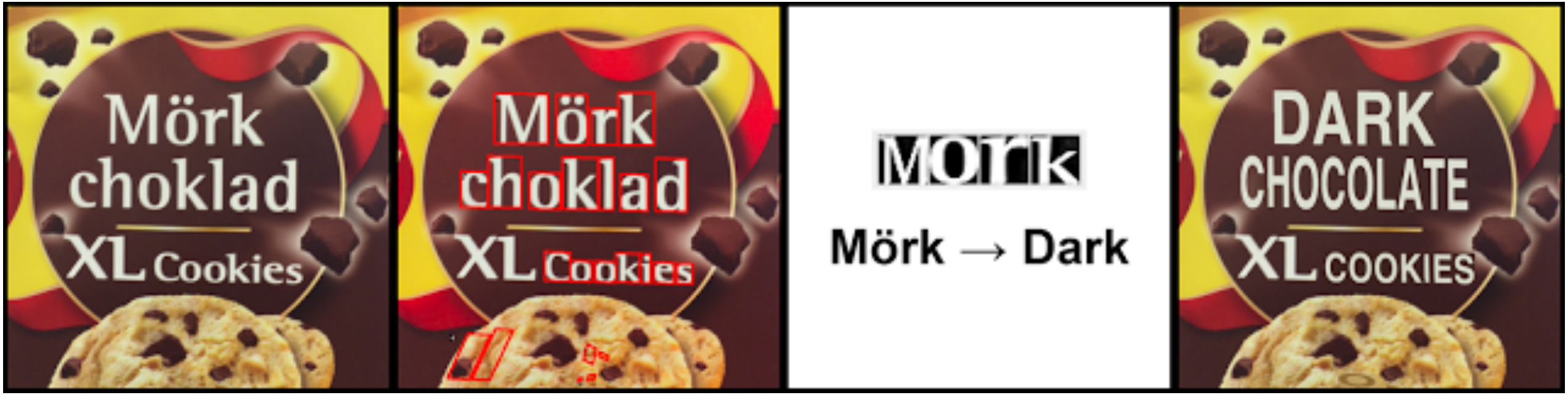
Shakespeare's Style Passage Generation

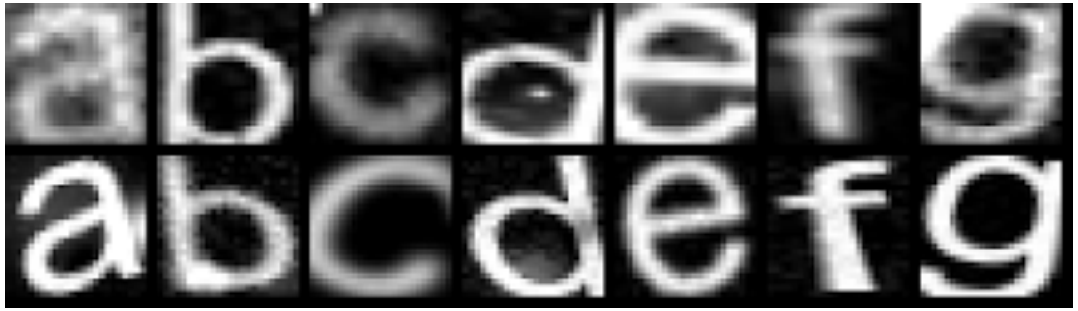
Remember, all the RNN knows are characters, so in particular it samples both speaker's names and the contents. Sometimes we also get relatively extended monologue passages, such as:

- VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.
- KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

Courtesy: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Machine Translation

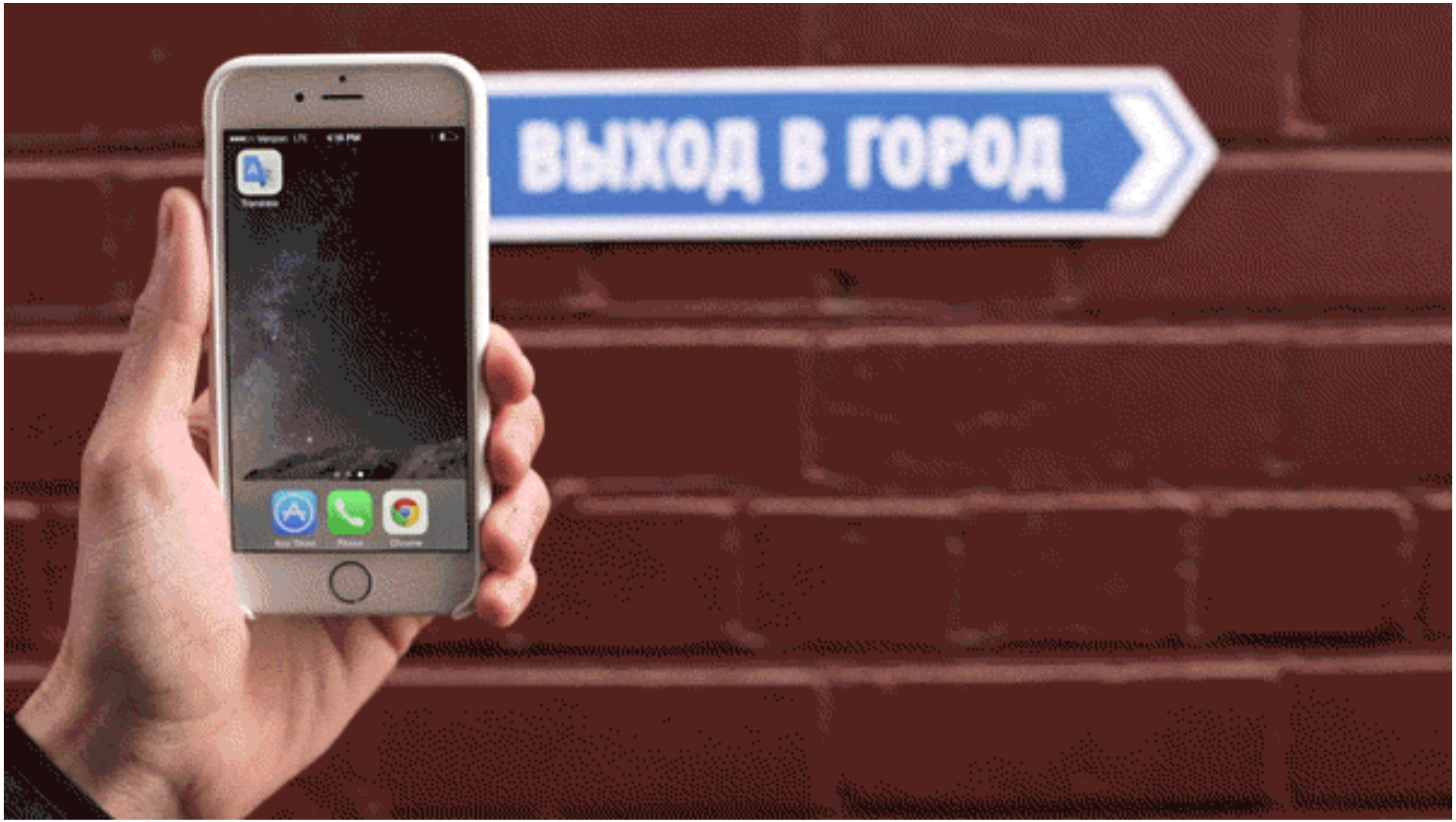




Some of the “dirty” letters we use for training. Dirt, highlights, and rotation, but not too much because we don’t want to confuse our neural net.

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

Google Translate

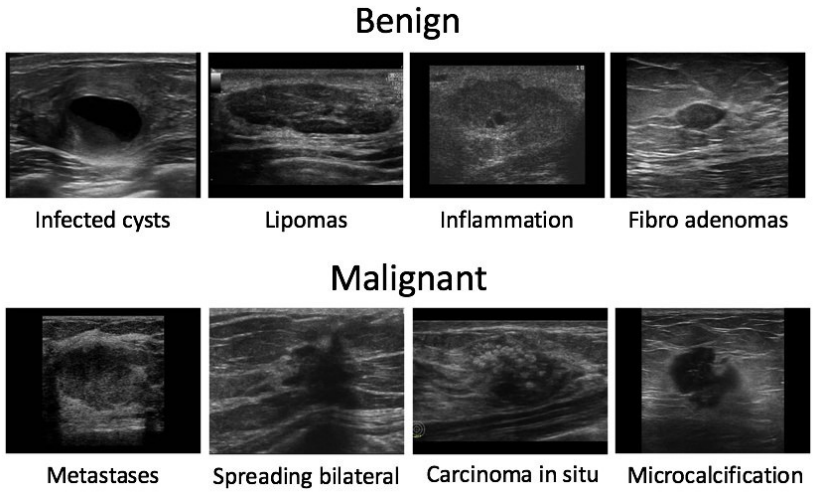


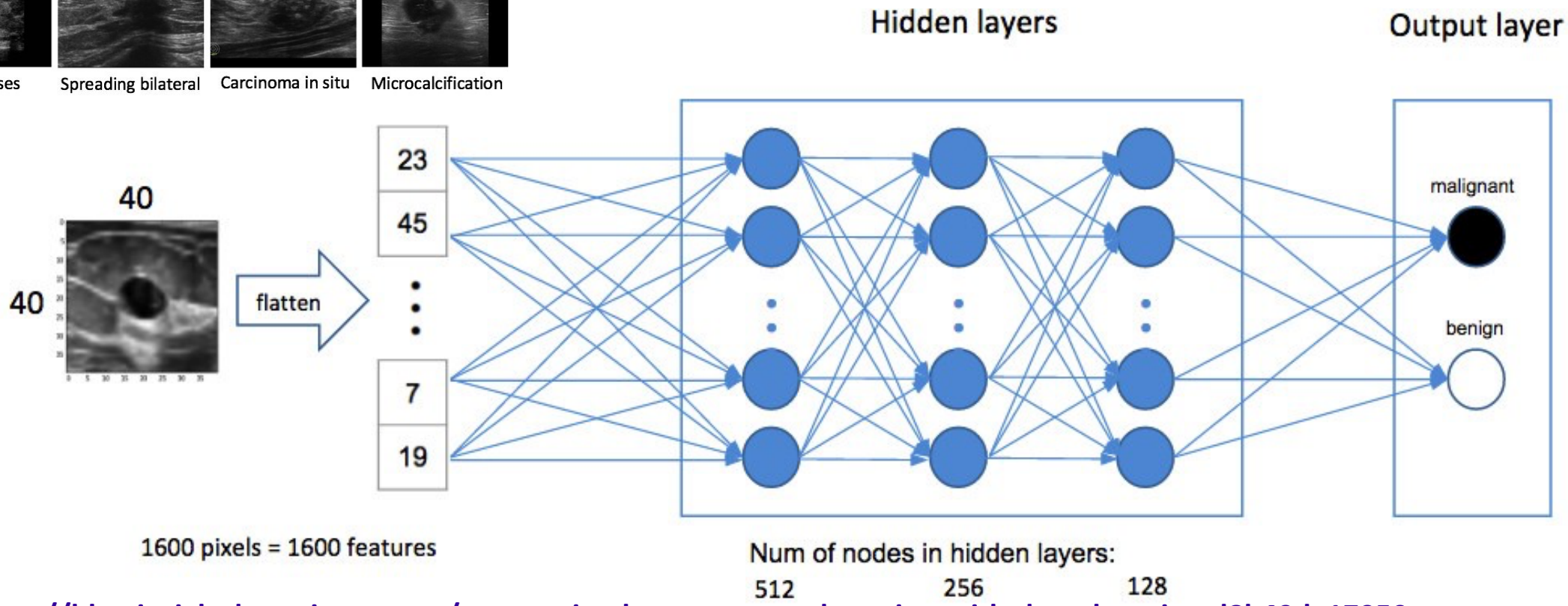
Courtesy: https://www.theverge.com/2015/1/14/7544919/google-translate-update-real-time-signs-conversations

Self Driving Cars

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

Cancer Detection





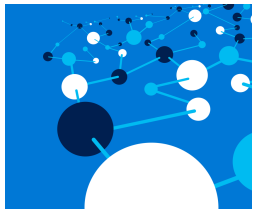
Courtesy: https://blog.insightdatascience.com/automating-breast-cancer-detection-with-deep-learning-d8b49da17950

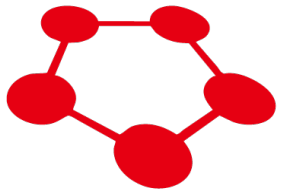
Outline

- **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
- Case Studies and Demos
- Open Issues and Challenges
- Conclusion

Deep Learning Frameworks

- Many Deep Learning frameworks!!
  - Berkeley Caffe
  - Facebook Caffe2
  - Google TensorFlow
  - Microsoft CNTK
  - Facebook Torch/PyTorch
  - Chainer/ChainerMN
  - Several Others …

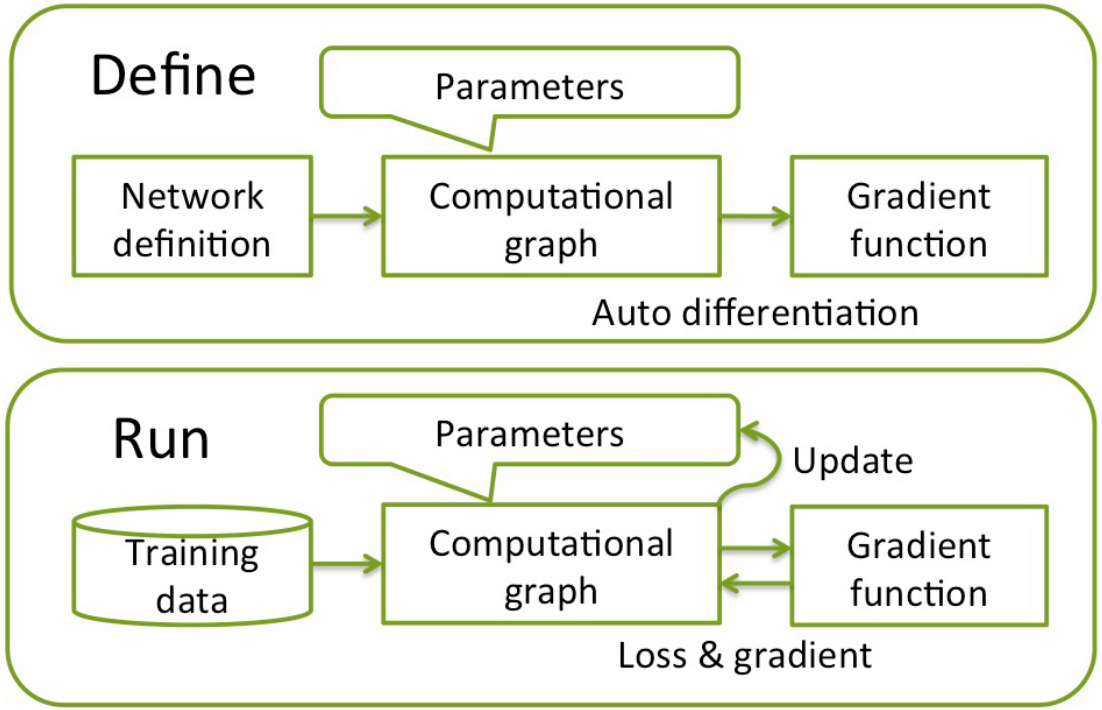


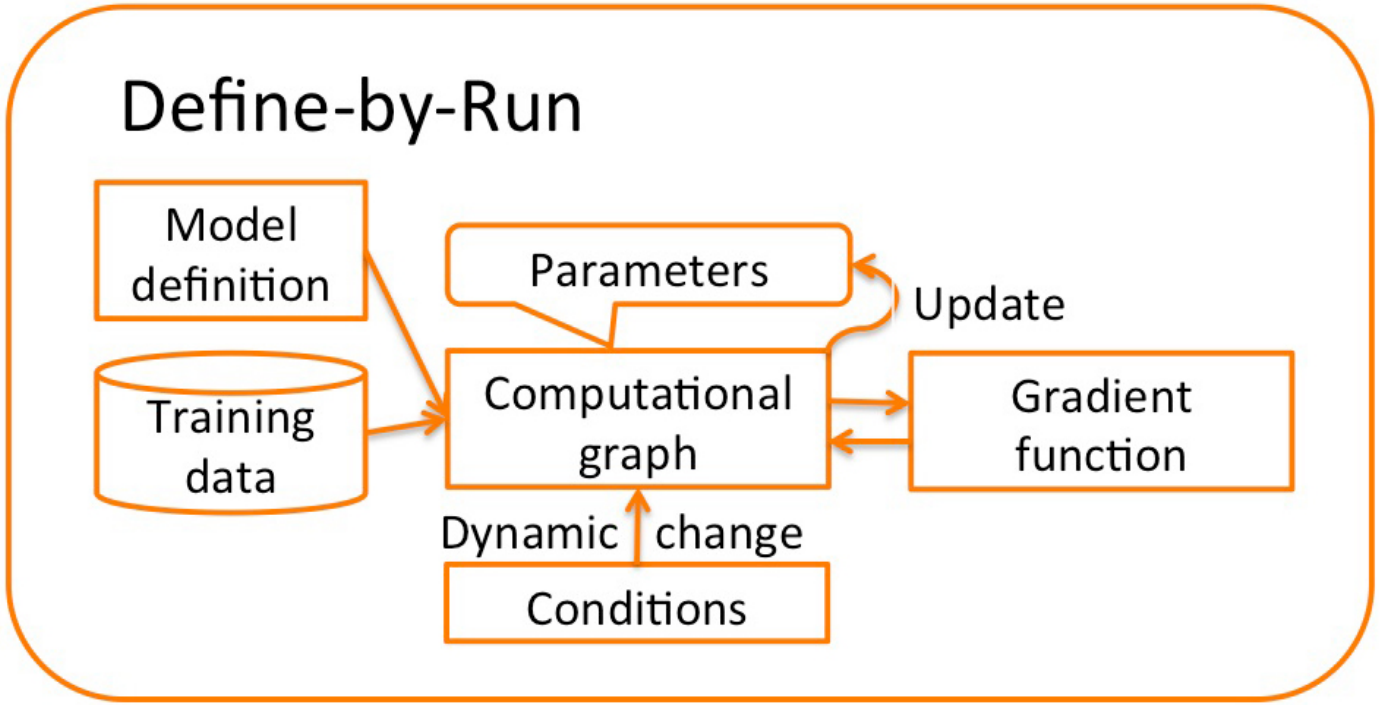


Why we need DL frameworks?

- Deep Learning frameworks have emerged
  - hide most of the *nasty mathematics*
  - focus on the *design* of neural networks
- Future DL frameworks are being designed in a distributed fashion
  - 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
  1. The HPC Eco-system: MPI-based Deep Learning
  2. Enterprise Eco-system: BigData-based Deep Learning

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

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

Berkeley (BVLC) Caffe

- One of the most popular DL framework
- Yangqing Jia (BVLC)
  - Author of Caffe and Caffe2 (Facebook)
- The framework has a modular C++ backend
- C++ and Python frontends
- Caffe is a single-node but multi-GPU framework
- Widely used for vision-based applications

Caffe

Courtesy: http://caffe.berkeleyvision.org

Berkeley (BVLC) Caffe

- Nearly 4,000 citations, usage by award papers at CVPR/ECCV/ICCV, and tutorials at ECCV'14 and CVPR'15
- Winner of the ACM MM open source award 2014
- Community: 250+ contributors, 15k+ subscribers on Github, and 7k+ members of the mailing list (as of August ‘17)
- Several efforts towards parallel/distributed training
  - OSU-Caffe - http://hidl.cse.ohio-state.edu/overview/
  - Intel-Caffe - https://github.com/intel/caffe
  - NVIDIA-Caffe - https://github.com/nvidia/caffe

Facebook Caffe2

- Caffe2 is a more versatile, diversified, and refactored framework
- Supported by Facebook
- Works on several platforms including mobile platforms
- https://github.com/caffe2/caffe2
- Main motivation
  - New Application Areas
  - Flexibility
  - Newer Platforms
    - Mobile

Courtesy: https://caffe2.ai

Facebook Caffe2

- Official Parallel/Distributed Training support
- Modularity: Multiple communication back-ends supported
  - Facebook Gloo (Redis/MPI to bootstrap communication)
  - NVIDIA NCCL
  - Message Passing Interface (MPI)
- Very recently launched – (All statistics as of August ’17)
  - 4 releases so far
  - 112 contributors
  - 2,300 commits
  - ~5000 GitHub stars

Google TensorFlow

- 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

Google TensorFlow

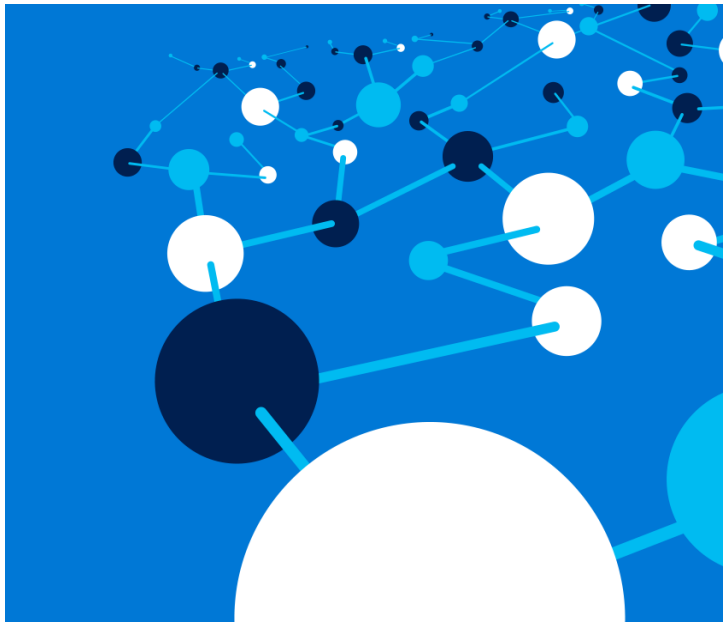
- As of August ’17 -- 21,000 commits and counting…
- 38 releases, 1,012 contributors, 67,000 GitHub stars, 33,000 forks
- Parallel/Distributed training
  - Official support through gRPC[1] library

- Several community efforts (TensorFlow/contrib)
  - MPI version by PNNL (MATEX) - https://github.com/matex-org/matex
  - MPI version by Baidu - https://github.com/baidu-research/tensorflow-allreduce
  - MPI+gRPC version by Minds.ai - https://www.minds.ai

[1] https://grpc.io/

Microsoft Cognitive Toolkit (CNTK)

- Formerly CNTK, now called the Cognitive Toolkit
- C++ backend
- C++ and Python frontend
- ASGD, SGD, and several others choices for Solvers/Optimizers
- Constantly evolving support for multiple platforms
- Performance has always been the “key feature”
- https://github.com/microsoft/cntk



Courtesy: https://www.microsoft.com/en-us/cognitive-toolkit/

Microsoft Cognitive Toolkit (CNTK)

- As of August ’17 – 14,000 commits
- 500 development branches and 146 contributors
- 12,000 GitHub stars and 3,000 forks
- Parallel and Distributed Training
  - MPI and NCCL support
- Community efforts
  - OSU’s CUDA-Aware CNTK*

* Dip Sankar Banerjee, Khaled Hamidouche, and Dhabaleswar K. Panda. ”Re-designing CNTK Deep Learning Framework on Modern GPU Enabled Clusters”, 8th IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Luxembourg 12-15, December 2016

Facebook Torch/PyTorch

- 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 plans to merge the PyTorch and Caffe2 efforts
- Key selling point is ease of expression and "define-by-run" approach

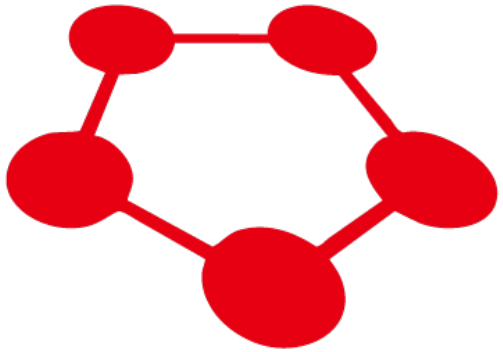
Courtesy: http://pytorch.org

Facebook Torch/PyTorch

- https://github.com/pytorch/pytorch
- Very active development (Statistics as of August ’17)
  - 4,200 commits
  - 287 contributors
  - 13 releases
  - ~7,000 GitHub stars
  - 1,300 forks
- Very recently got distributed training support
  - http://pytorch.org/docs/master/distributed.html

Preferred Networks Chainer/ChainerMN

- Very recent addition to the plethora of DL frameworks
- Preferred Networks (PN) is an NVIDIA Inception Program Startup
- Chainer is an emerging framework
- Uses ***Define-by-run*** (Chainer, PyTorch) approach instead of ***Define-and-run*** (Caffe, TensorFlow, Torch, Theano) approach
- https://github.com/chainer/chainer
  - As of August '17 – 48 releases already
  - 10,455 commits
  - 134 contributors
  - 2,853 stars and 753 forks on Github



Chainer/Chainer MN

- 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
  - 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 Neon (Nervana) - https://github.com/NervanaSystems/neon
- 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

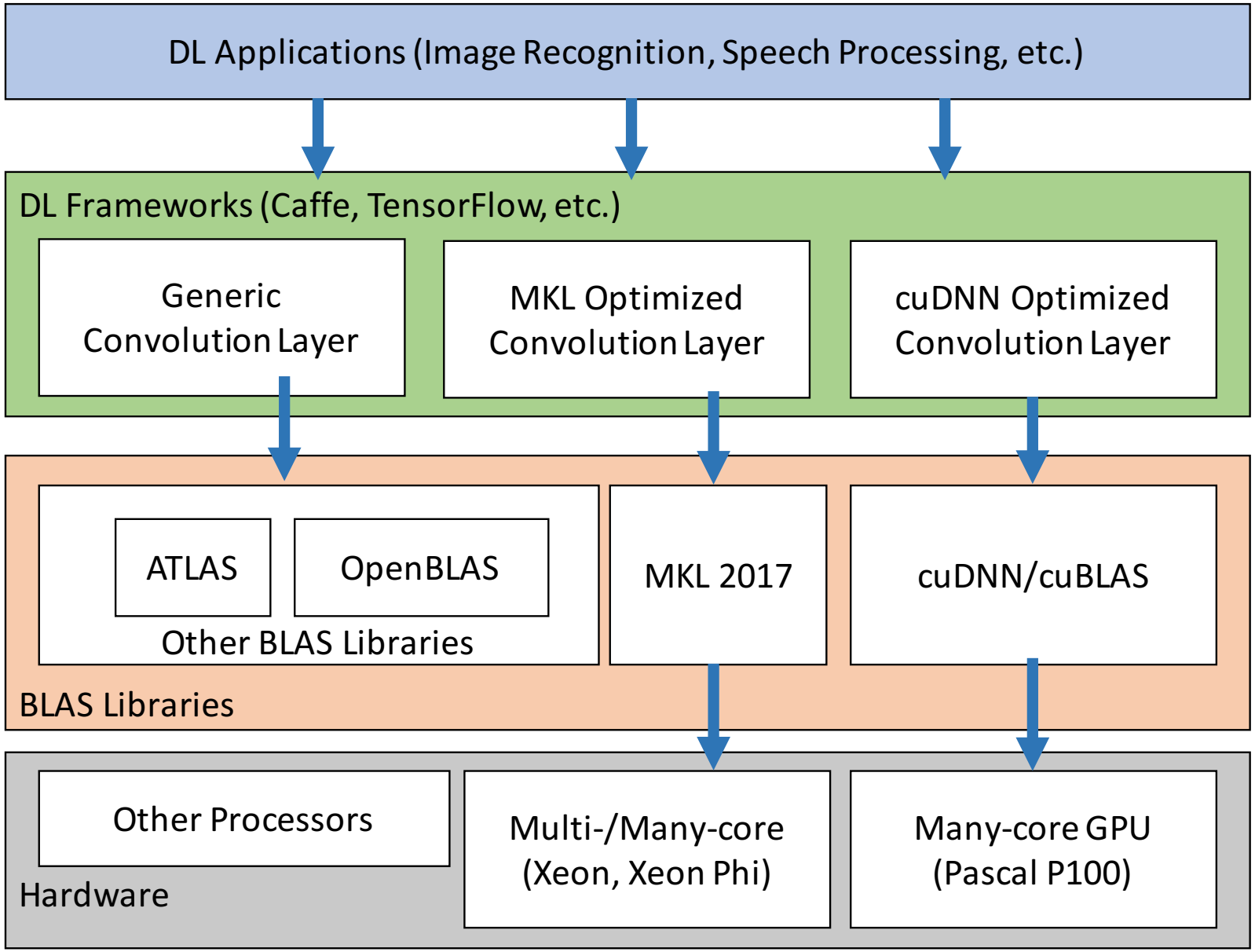
- Introduction
- **Overview of Execution Environments**
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- Case Studies and Demos
- Open Issues and Challenges
- Conclusion

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
    - Very recently, Google announced Deeplearn.js (a DL framework in a web-browser)

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

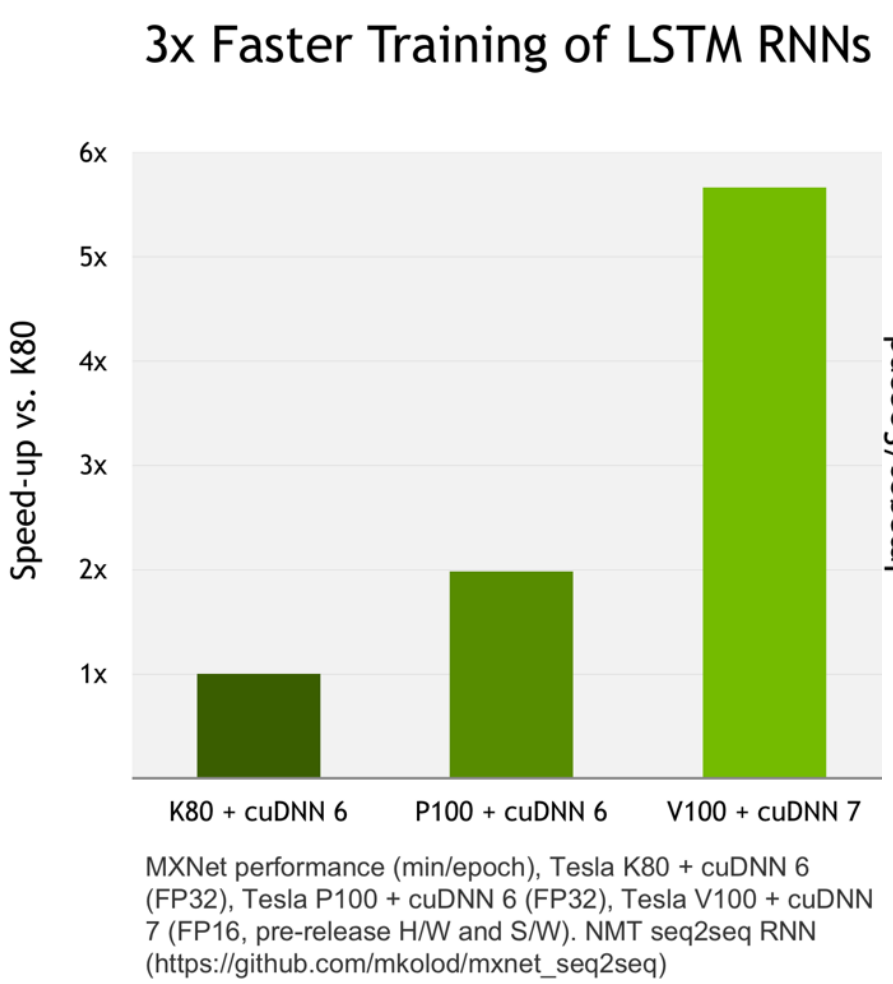


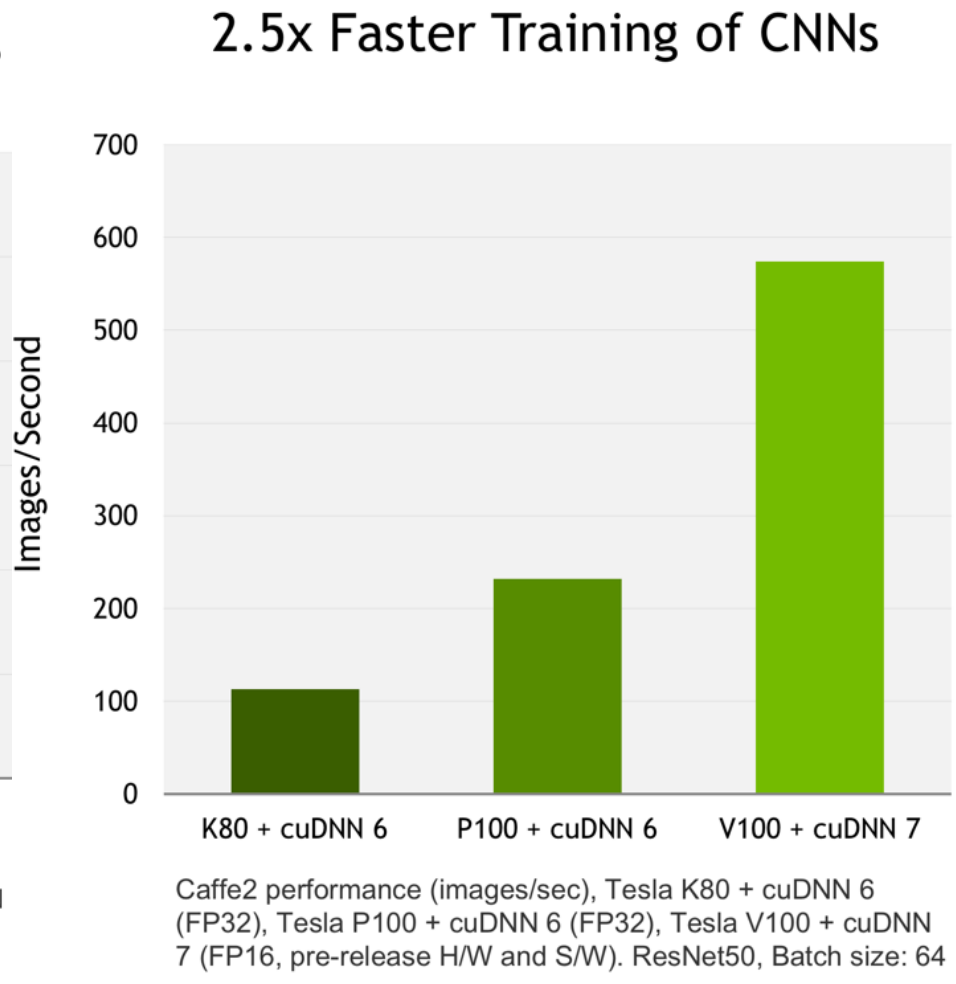
DL Frameworks and Underlying Libraries

- 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)
  - Intel MKL-DNN (MKL 2017) – recent but a very promising development

Where does the Performance come from?

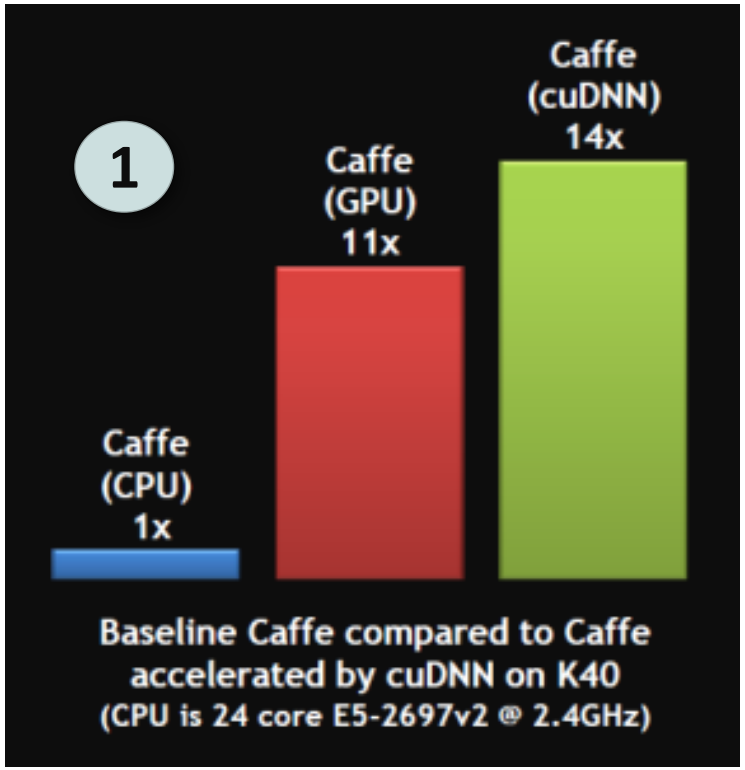
- Performance Improvements can be observed because of:
  - Faster convolutions with each successive cuDNN version
  - Faster hardware and more FLOPS as we move from:
    K-80 -> P-100 -> V-100

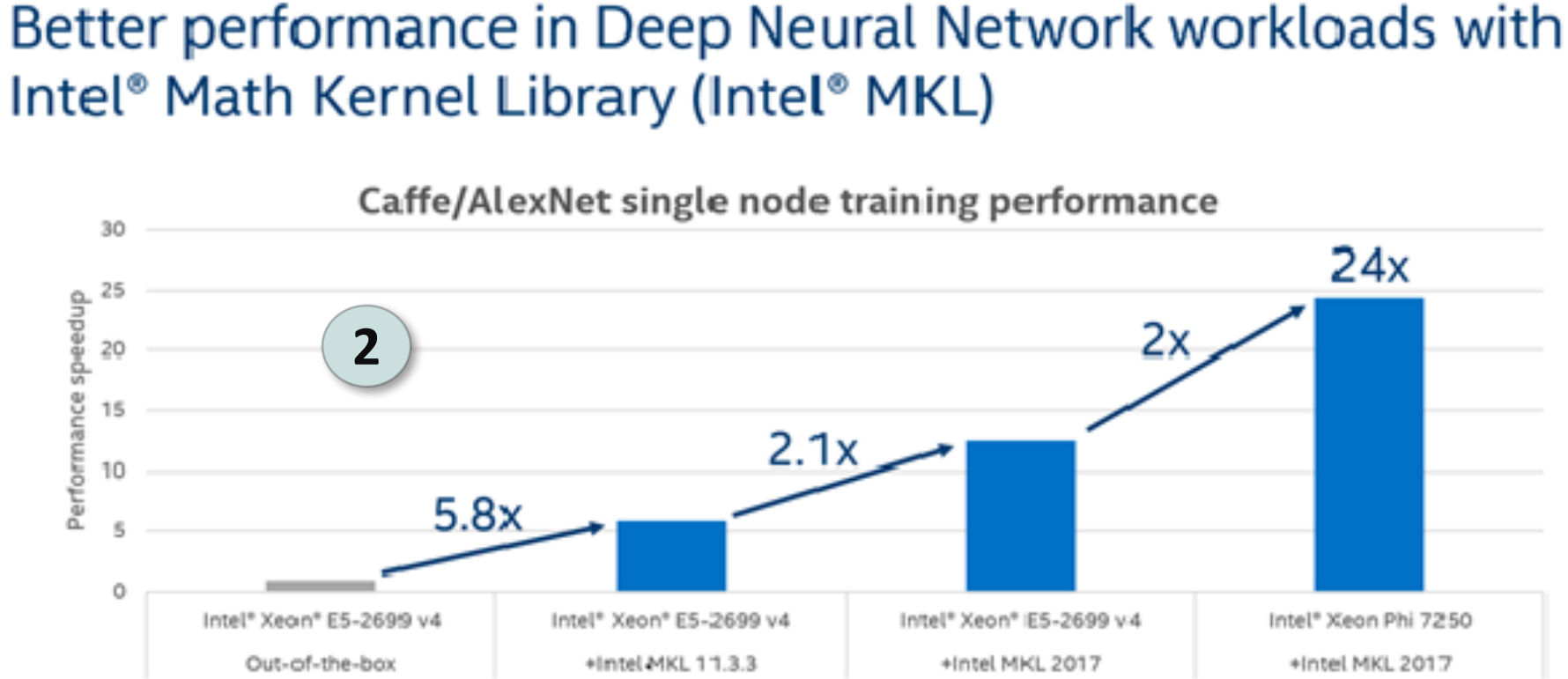




Courtesy: https://developer.nvidia.com/cudnn

Differences in How Vendors Report Performance





- NVIDIA reports performance [1] with the basic Caffe version (no multi-threading and no optimized MKL support)
- Intel reports [2] performance gains over the same basic Caffe version
- *Hence, the need for a holistic and fair comparison!!*

1. https://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library/
2. https://software.intel.com/en-us/articles/introducing-dnn-primitives-in-intelr-mkl

Outline

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

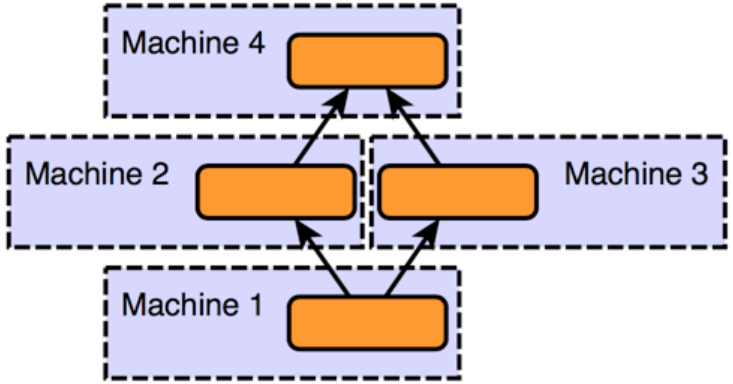
The Need for Parallel and Distributed Training

- Why do we need Parallel Training?
- Larger and Deeper models are being proposed; **AlexNet** to **ResNet** to **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!!**

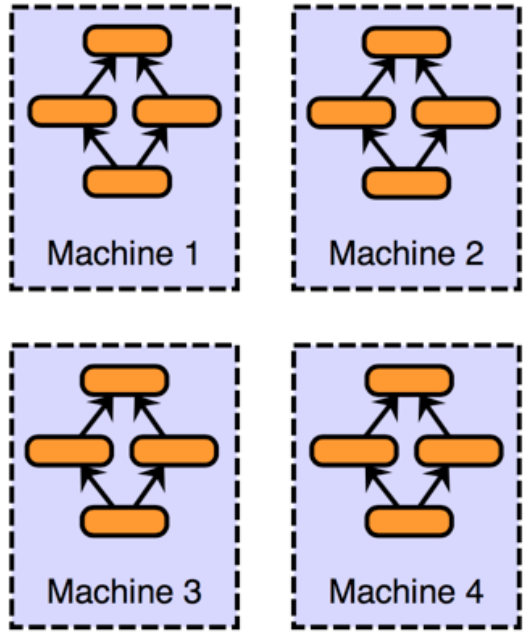
Parallelization Strategies

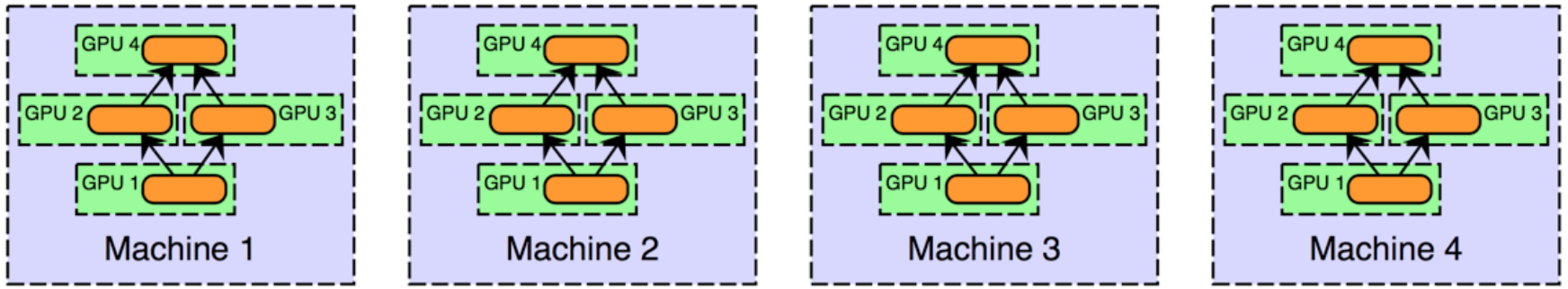
- What are the Parallelization Strategies
  - Model Parallelism
  - Data Parallelism
  - Hybrid Parallelism
  - Automatic Selection

Model Parallelism



Data Parallelism

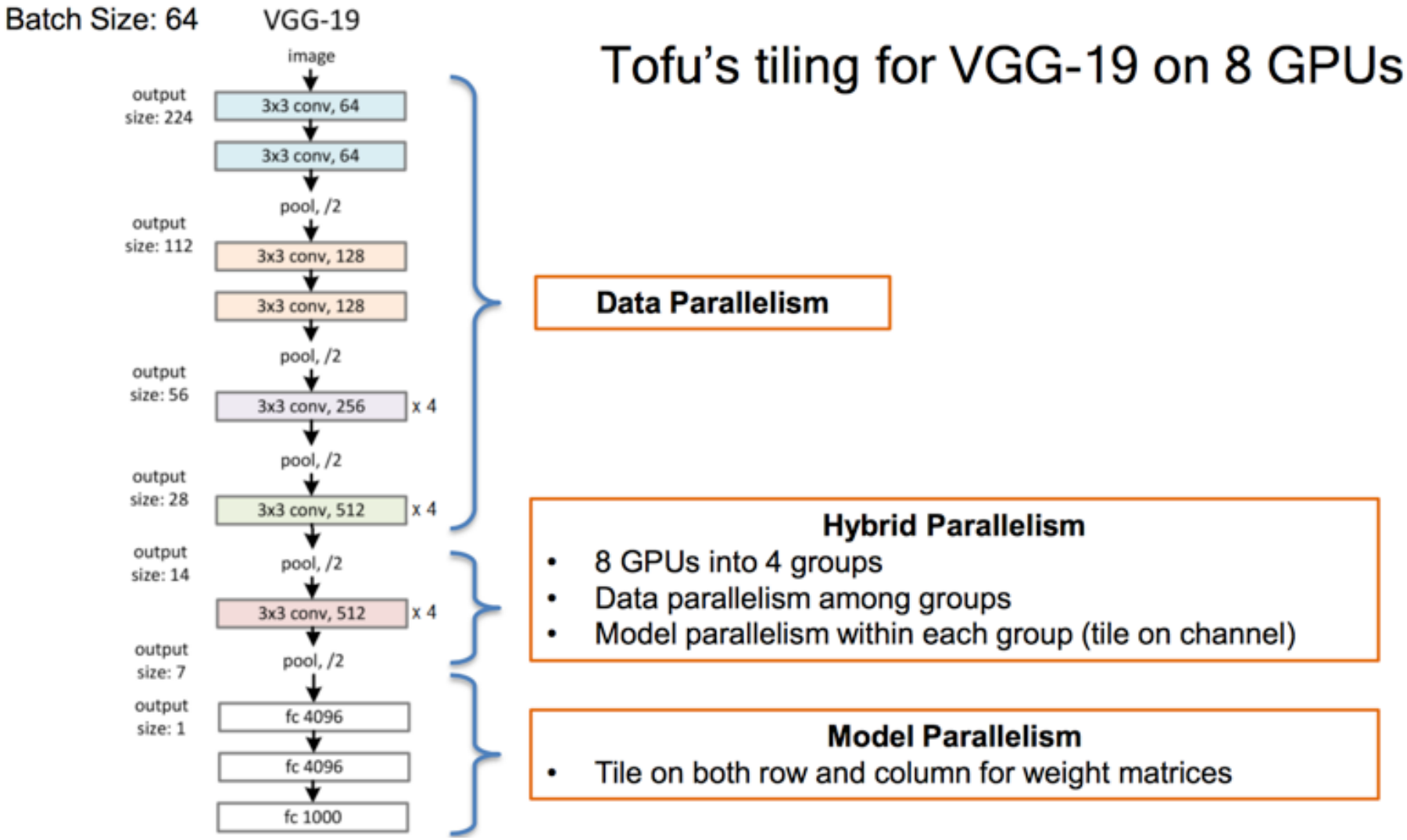




Hybrid (Model and Data) Parallelism

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

Automatic Selection of Parallelization Strategies



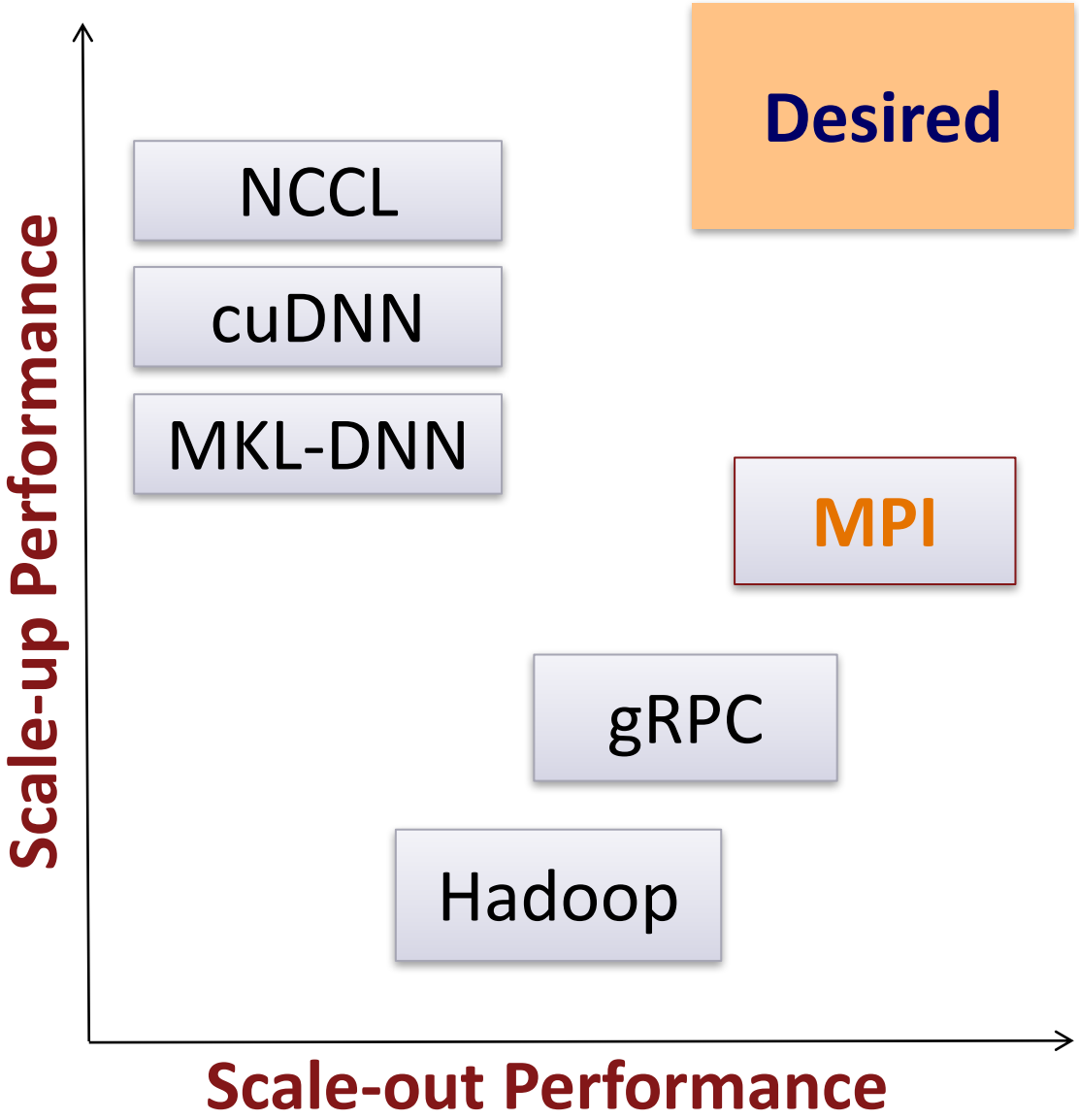
Courtesy: http://on-demand.gputechconf.com/gtc/2017/presentation/s7724-minjie-wong-tofu-parallelizing-deep-learning.pdf

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 (upcoming 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-based
    - Google TensorFlow – gRPC-based/MPI efforts
    - Facebook Caffe2 – Hybrid (NCCL/Gloo/MPI)



Outline

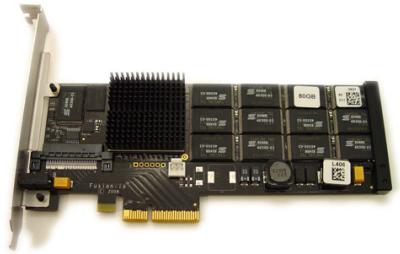
- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- **Latest Trends in HPC Technologies**
- Challenges in Exploiting HPC Technologies for Deep Learning
- Case Studies and Demos
- Open Issues and Challenges
- Conclusion

Drivers of Modern HPC Cluster Architectures

Multi-core Processors

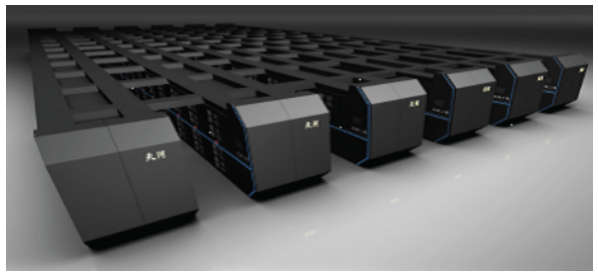
High Performance Interconnects - InfiniBand
<1usec latency, 100Gbps Bandwidth>

Accelerators / Coprocessors
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 and Intel Xeon Phi)



Tianhe – 2

Titan

Stampede

Tianhe – 1A

HPC Technologies

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

Overview of High Performance Interconnects

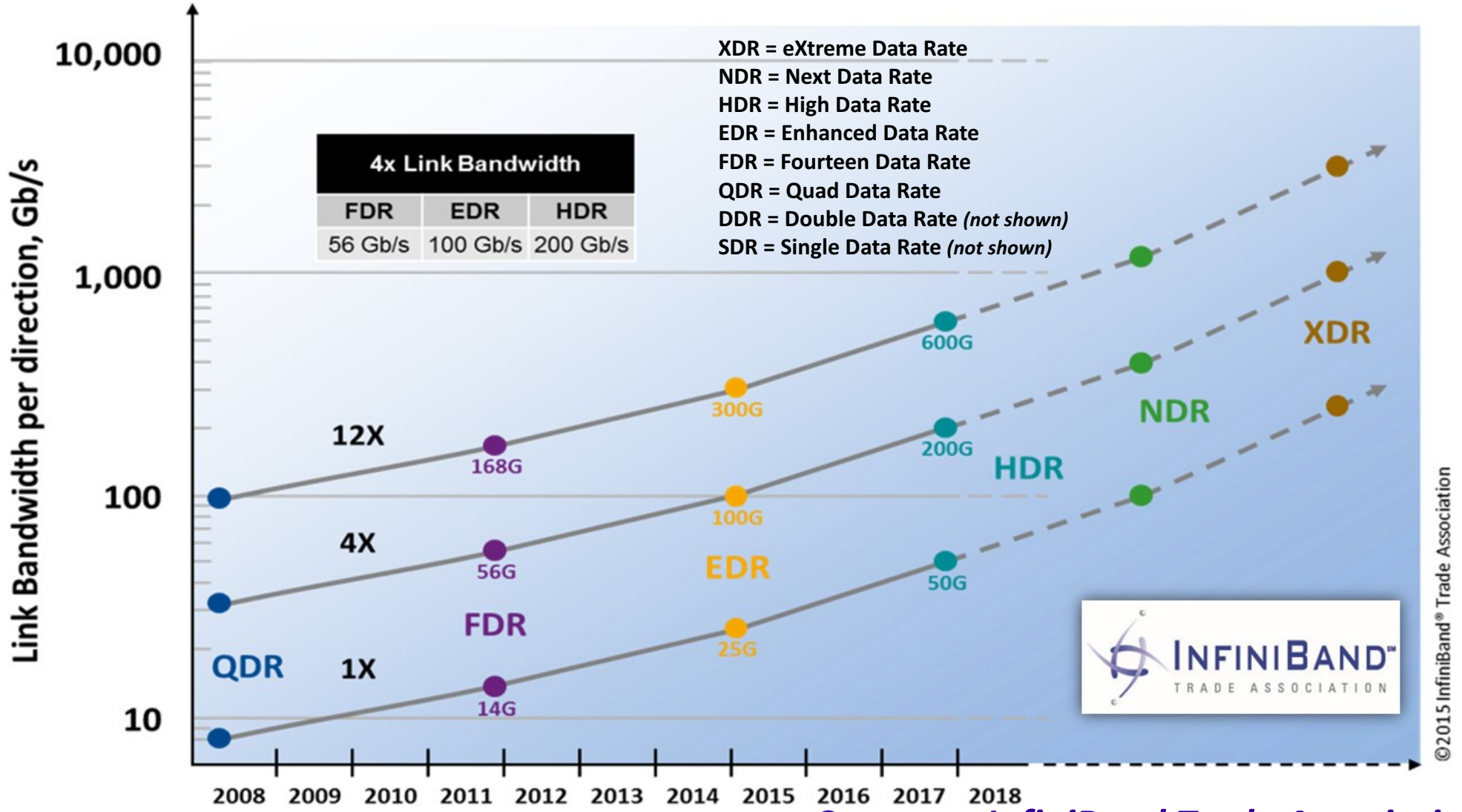
- High-Performance Computing (HPC) has adopted advanced interconnects and protocols
  - InfiniBand (IB)
  - Omni-Path
  - High Speed Ethernet 10/20/40/100 Gigabit Ethernet/iWARP
  - RDMA over Converged Enhanced Ethernet (RoCE)
- Very Good Performance
  - Low latency (few micro seconds)
  - High Bandwidth (100 Gb/s with EDR 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 (2016 -)	200 Gbit/sec (4X HDR)

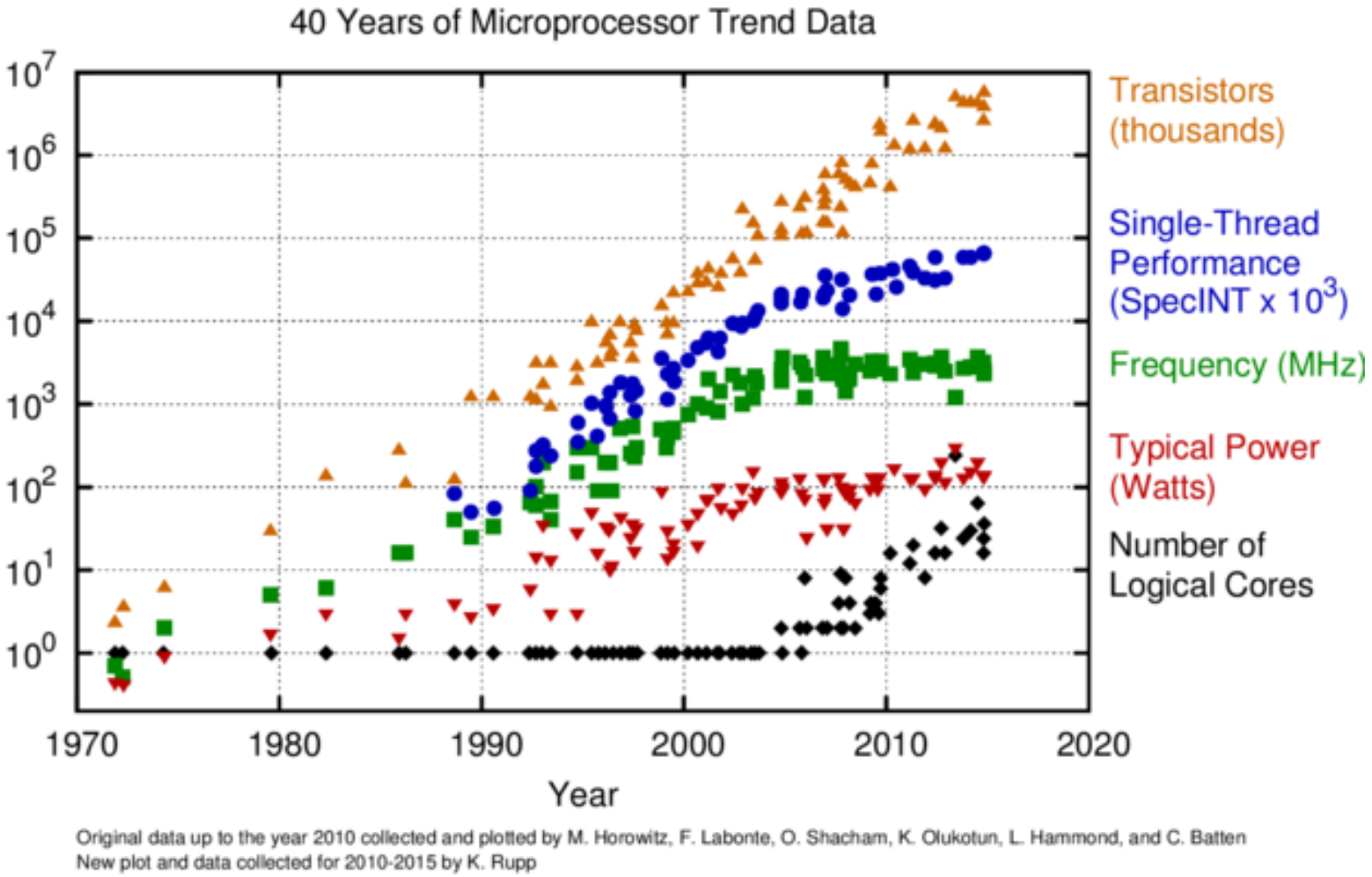
100 times in the last 15 years

InfiniBand Link Speed Standardization Roadmap



Courtesy: InfiniBand Trade Association

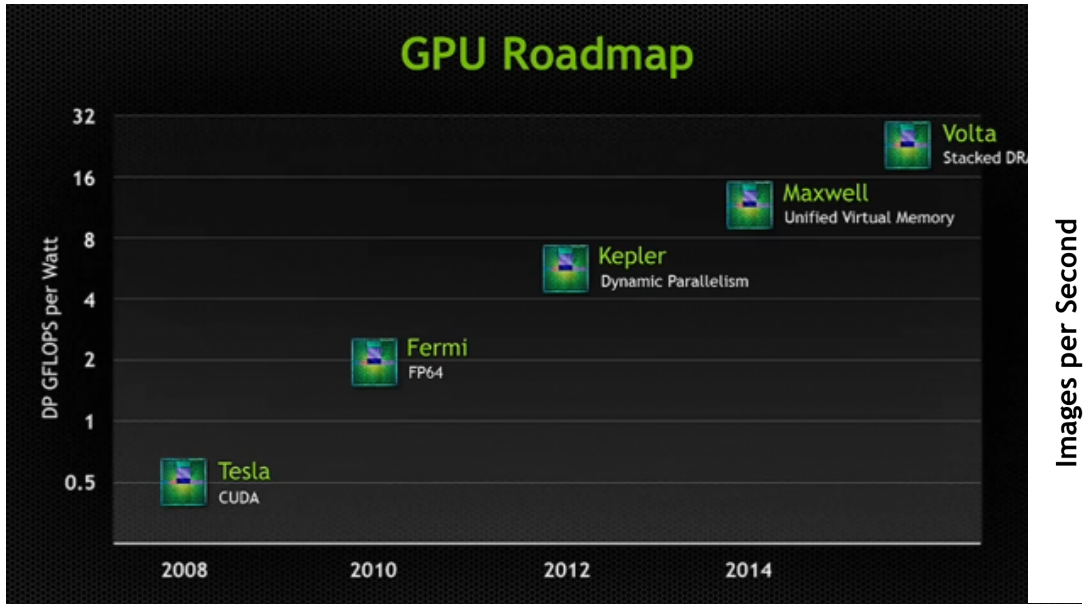
Trends in Microprocessor Technology

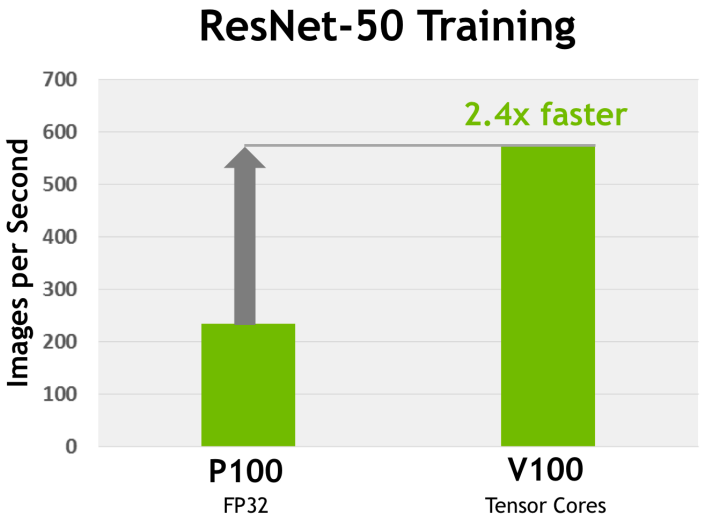


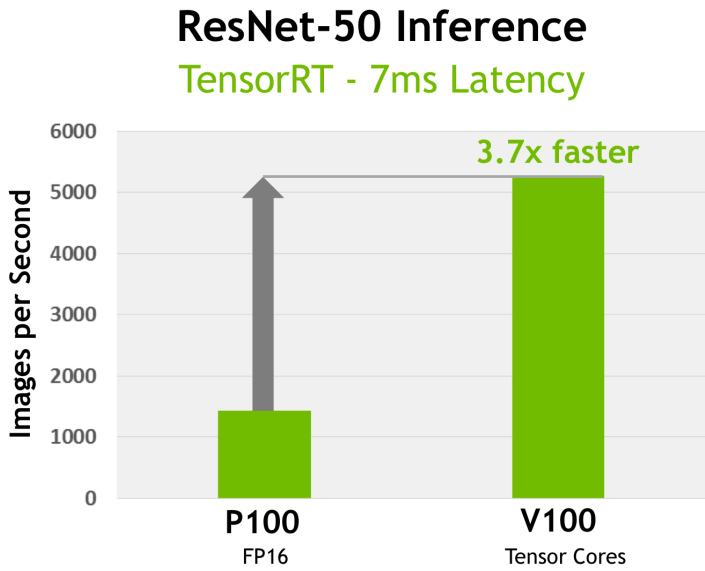
- Small, yet steady increase in single thread performance
- Rapid increase in number of transistors per chip
- Power consumption has remained more or less constant
- Rapid increase in number of cores

Courtesy: https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/

Trends in GPU Technology





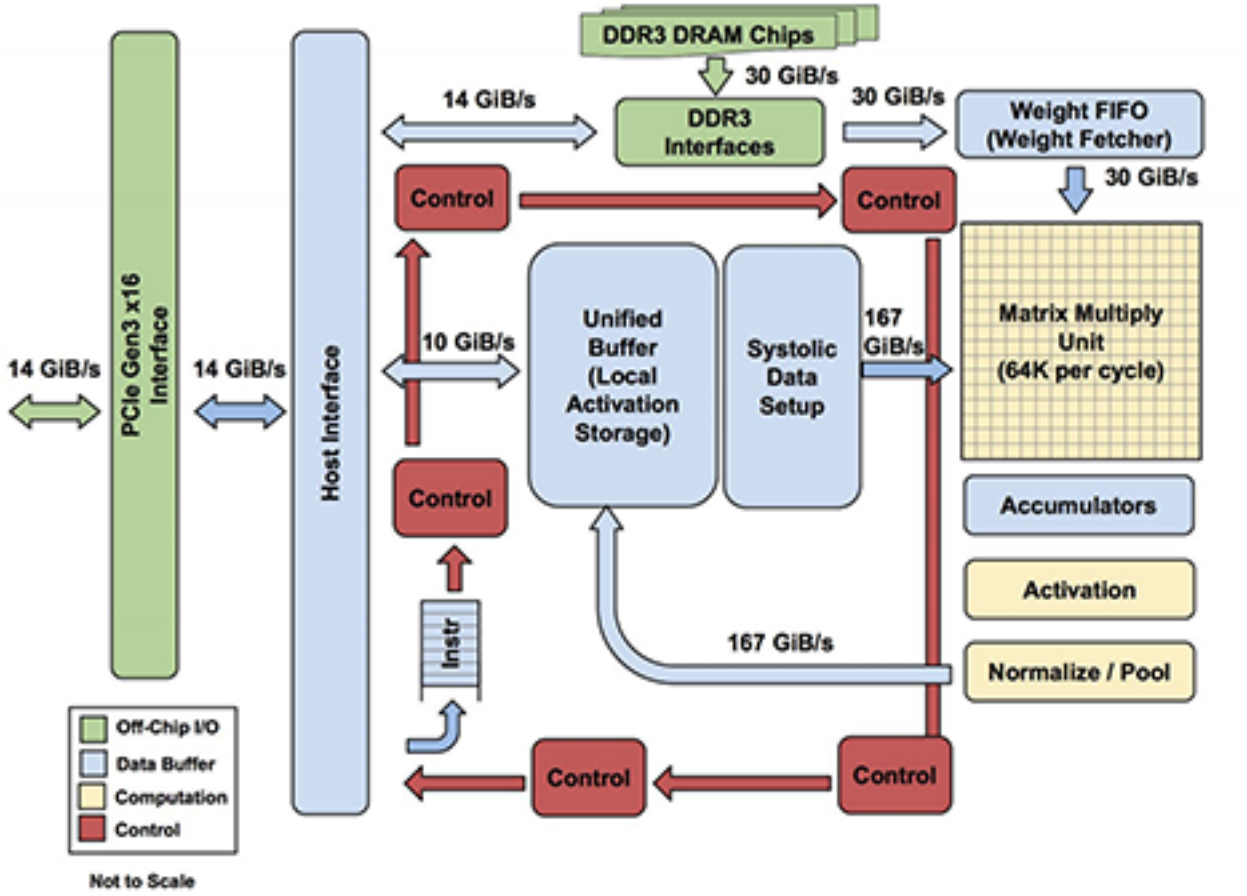


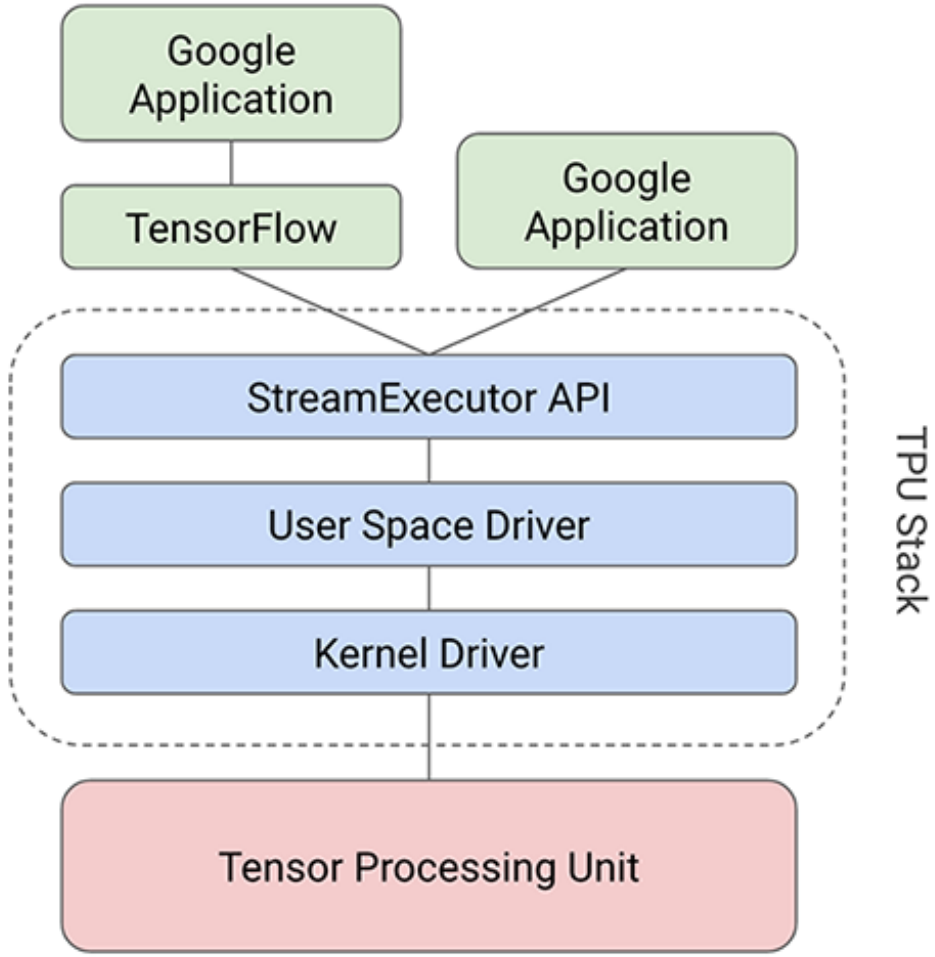
- NVIDIA Volta is optimized for Deep Learning workloads
  - has dedicated "Tensor Cores" for both Training and Inference
  - 2.4X faster than Pascal GPUs for ResNet-50 training

Courtesy: https://devblogs.nvidia.com/parallelforall/inside-volta/

: http://wccftech.com/nvidia-roadmap-2017-update-volta-gpu/

Google TPU





- CISC style instruction set
- Uses systolic arrays as the heart of multiply unit

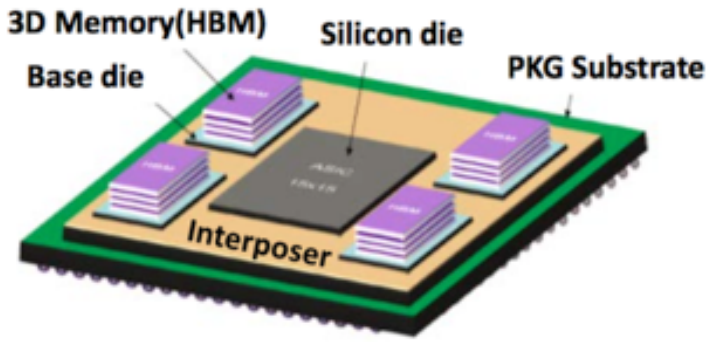
Courtesy: https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

: https://www.nextplatform.com/2017/04/05/first-depth-look-googles-tpu-architecture/

Intel Nervana also has a TPU

nervana tensor processing unit

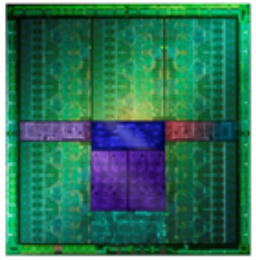
- Unprecedented compute density
- Scalable distributed architecture
- Memory near computation
- Learning and inference
- Exploit limited precision
- Incorporate latest advances
- Power efficiency



Model and substrate for computation

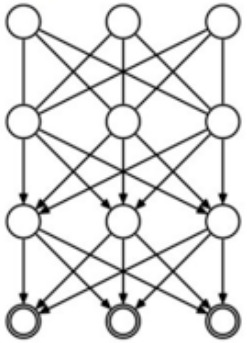
Do this instead:

Feasible, but still hard.



Custom ASIC

- Model description language
- Hardware abstraction layer
- Distributed primitives
- Compilers, drivers



Deep learning model

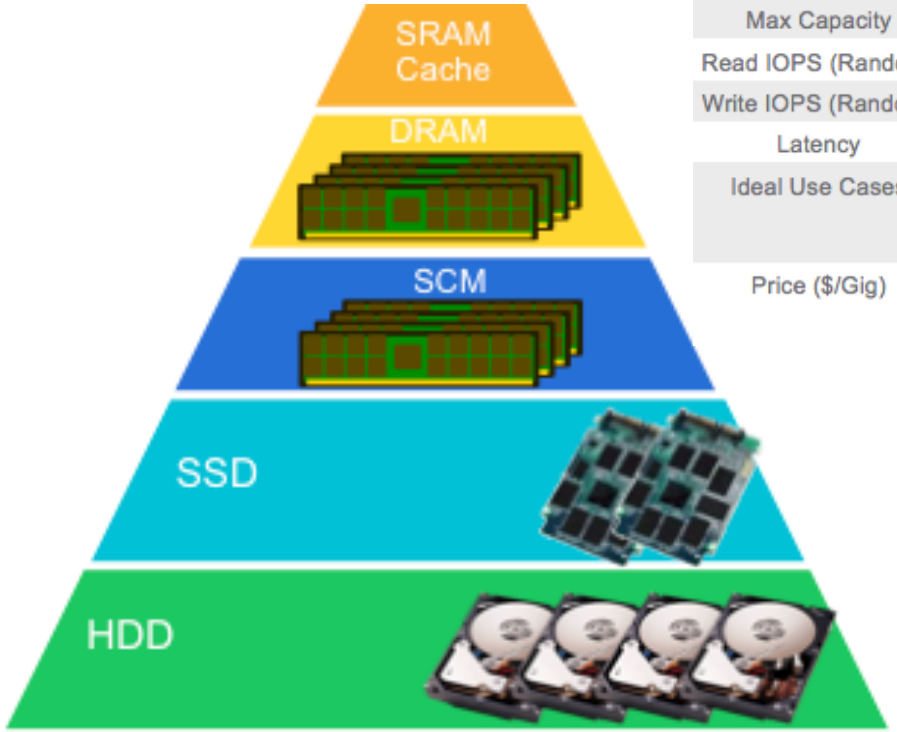
Courtesy: http://cra.org/ccc/wp-content/uploads/sites/2/2016/08/CCC-Nervana-Amir.pdf

Trends in High-Performance Storage

	NVMe	NVRAM	3D XPoint
Definition	High Speed interface for SSDs in a PCIe form factor used as **block storage**	**Non-volatile DRAM** backed up battery or super capacitor used as **byte addressable memory**	Non-volatile high performance (1000x NAND), high density (8-10X DRAM), high endurance (1000X NAND) **byte addressable memory**
Form Factor	Connects to PCIe bus	Connects to a DDR3 DIMM slot	Connects to a DDR3 DIMM slot
Max Capacity	2 TB	**16GB**	**128 GB**
Read IOPS (Random)	750,000	1.4 Million	In millions
Write IOPS (Random)	430,000	1.4 Million	In millions
Latency	15 Microsecond	10 Nanoseconds	10 Nanoseconds
Ideal Use Cases	Caching Tier: Transactional workloads **requiring high IOPS**	**Byte Addressable memory for metadata & client side caching, reduce write amplification**	**Highly Dense Byte Addressable memory for high speed caching, staging dedup/compression**
Price ($/Gig)	**$$**	**$$$**	**$$$$**

Memory

Storage



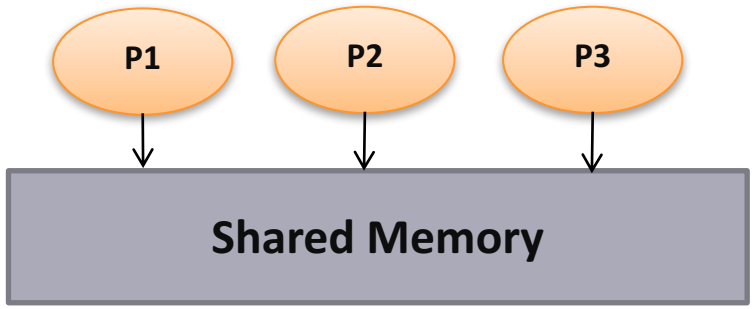
Courtesy: https://blogs.vmware.com/virtualblocks/2016/03/01/the-evolution-of-next-gen-hci-part-2-future-of-all-flash-virtual-san/

: https://www.rambus.com/blogs/mid-when-memory-and-storage-converge/

HPC Technologies

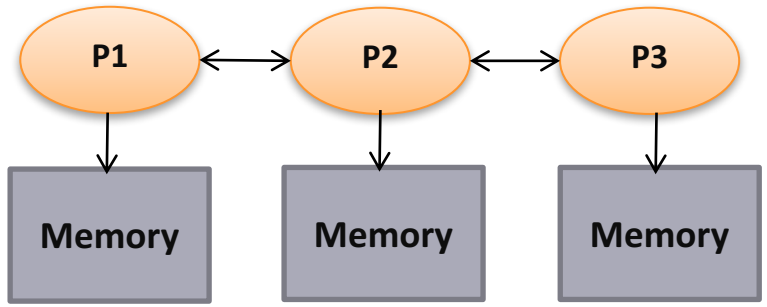
- Hardware
  - Interconnects – InfiniBand, RoCE, Omni-Path, etc.
  - Processors – GPUs, Multi-/Many-core CPUs, Tensor Processing Unit (TPU), FPGAs, etc.
  - Storage – NVMe, SSDs, Burst Buffers, etc.
- **Communication Middleware**
  - **Message Passing Interface (MPI)**
    - **CUDA-Aware MPI, Many-core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

Parallel Programming Models Overview



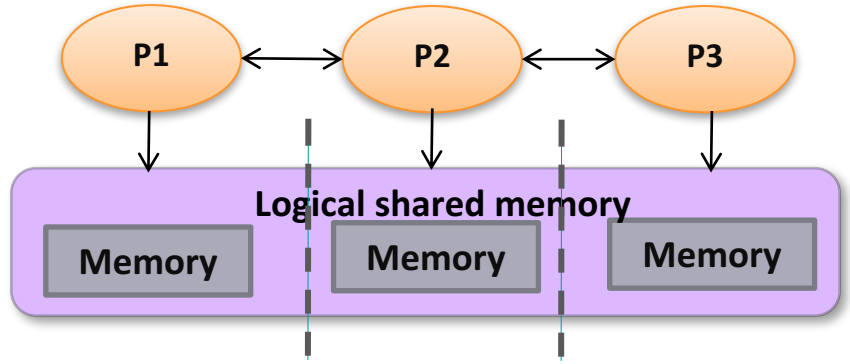
Shared Memory Model

SHMEM, DSM



Distributed Memory Model

MPI (Message Passing Interface)

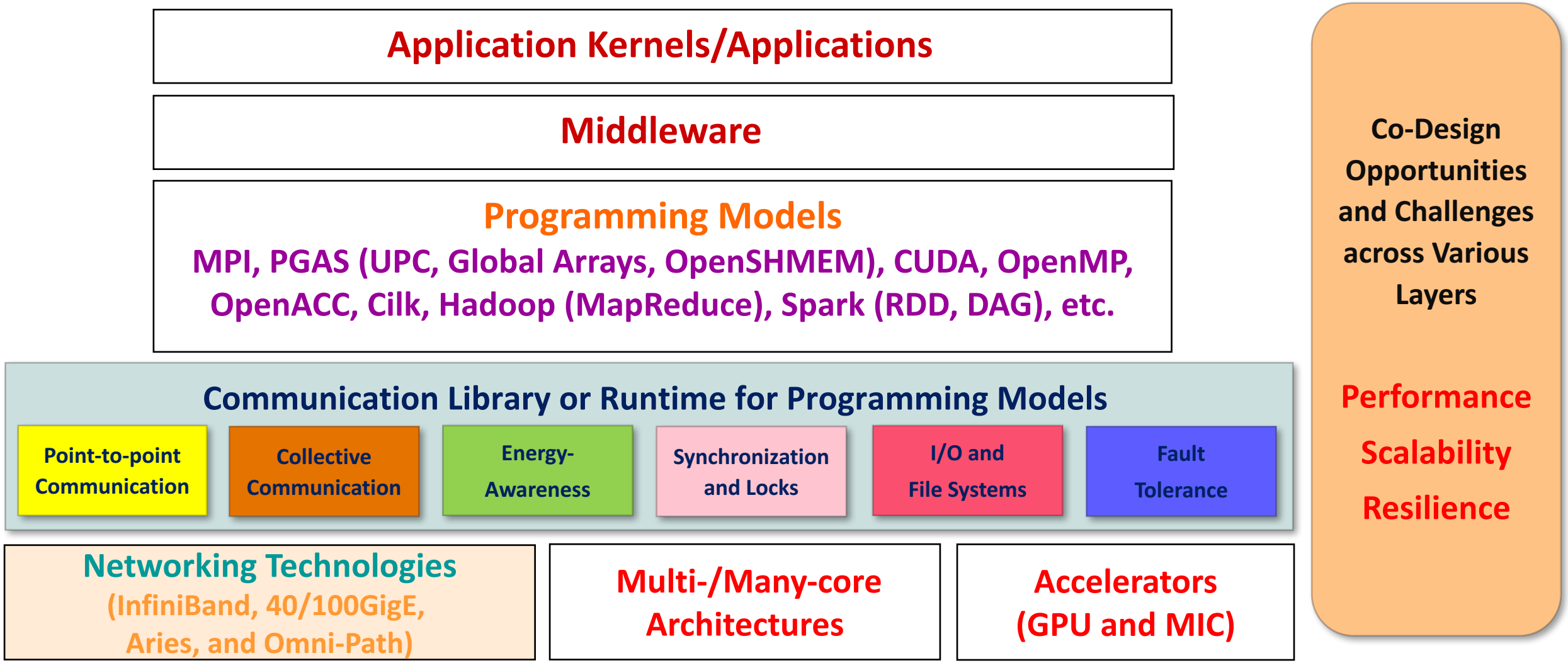


Partitioned Global Address Space (PGAS)

Global Arrays, 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

Supporting Programming Models for Multi-Petaflop and Exaflop Systems: Challenges



High-Performance Programming Model Implementations

- Message Passing Interface (MPI)
  - OpenMPI
  - IntelMPI
  - CrayMPI
  - IBM Spectrum MPI
  - MVAPICH2
  - And many more…
- Partitioned Global Address Space (PGAS)
  - Gasnet
  - CraySHMEM
  - MVAPICH2-X
  - And many more…

MVAPICH

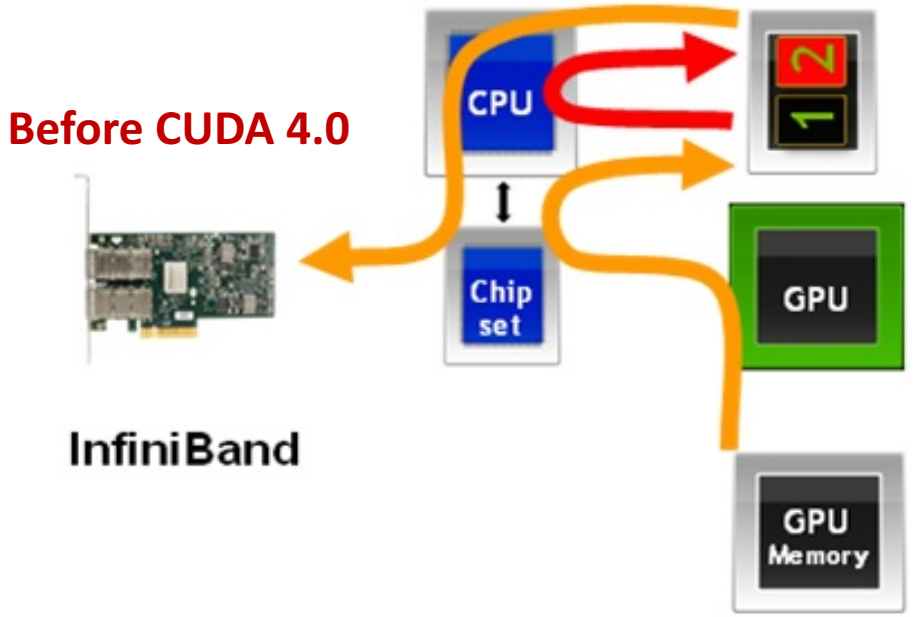
IBM

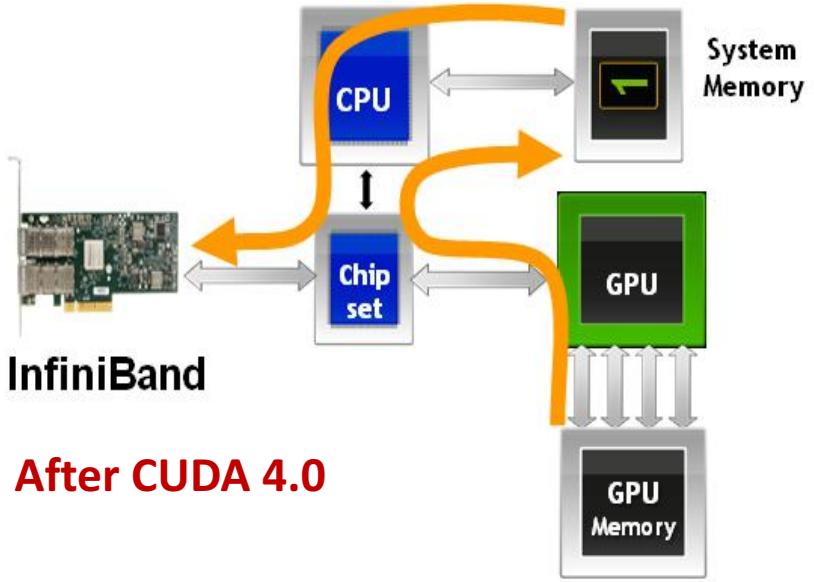
Overview of the MVAPICH2 Project

- 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-3.0), 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 2,800 organizations in 85 countries**
  - **More than 425,000 (> 0.4 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (June '17 ranking)
    - 1st, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China
    - 15th, 241,108-core (Pleiades) at NASA
    - 20th, 462,462-core (Stampede) at TACC
    - 44th, 74,520-core (Tsubame 2.5) at Tokyo Institute of Technology
  - Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)
  - **http://mvapich.cse.ohio-state.edu**
- Empowering Top500 systems for over a decade
  - System-X from Virginia Tech (3rd in Nov 2003, 2,200 processors, 12.25 TFlops) ->
  - Stampede at TACC (12th in Jun'16, 462,462 cores, 5.168 Plops)

CUDA Aware MPI (Before CUDA 5.0)

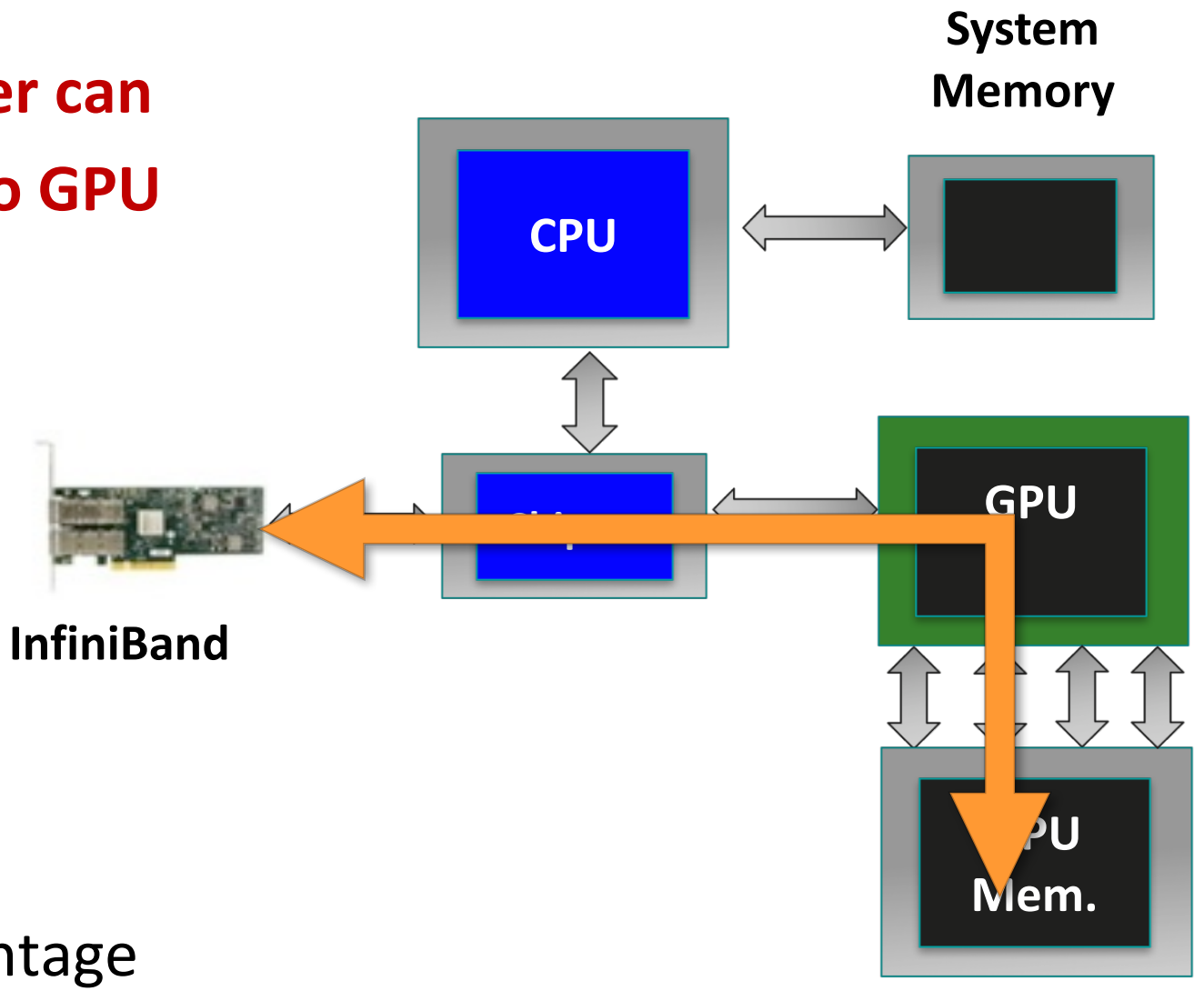
- Before CUDA 4.0, lack of a common memory registration mechanism
  - Each device has to pin the host memory
  - Many operating systems do not allow multiple devices to register the same memory pages
  - Previous solution: Use different buffer for each device and copy the data
- After CUDA 4.0, both devices register a common host buffer
  - GPU copies data to this buffer, and the network adapter can directly read from this buffer (or vice-versa)



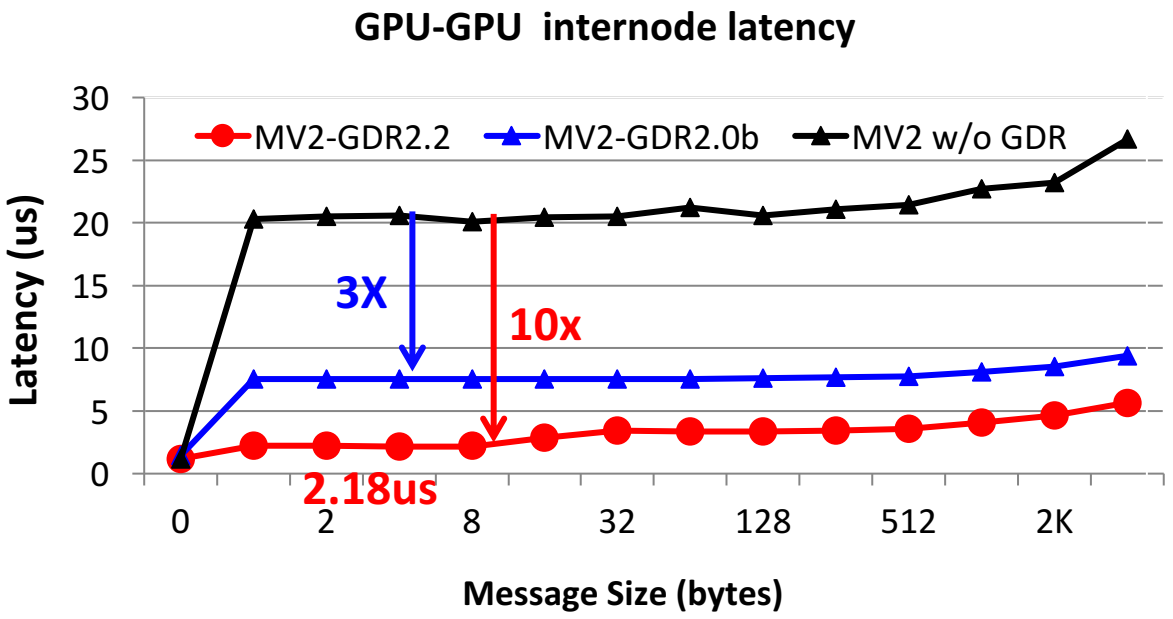


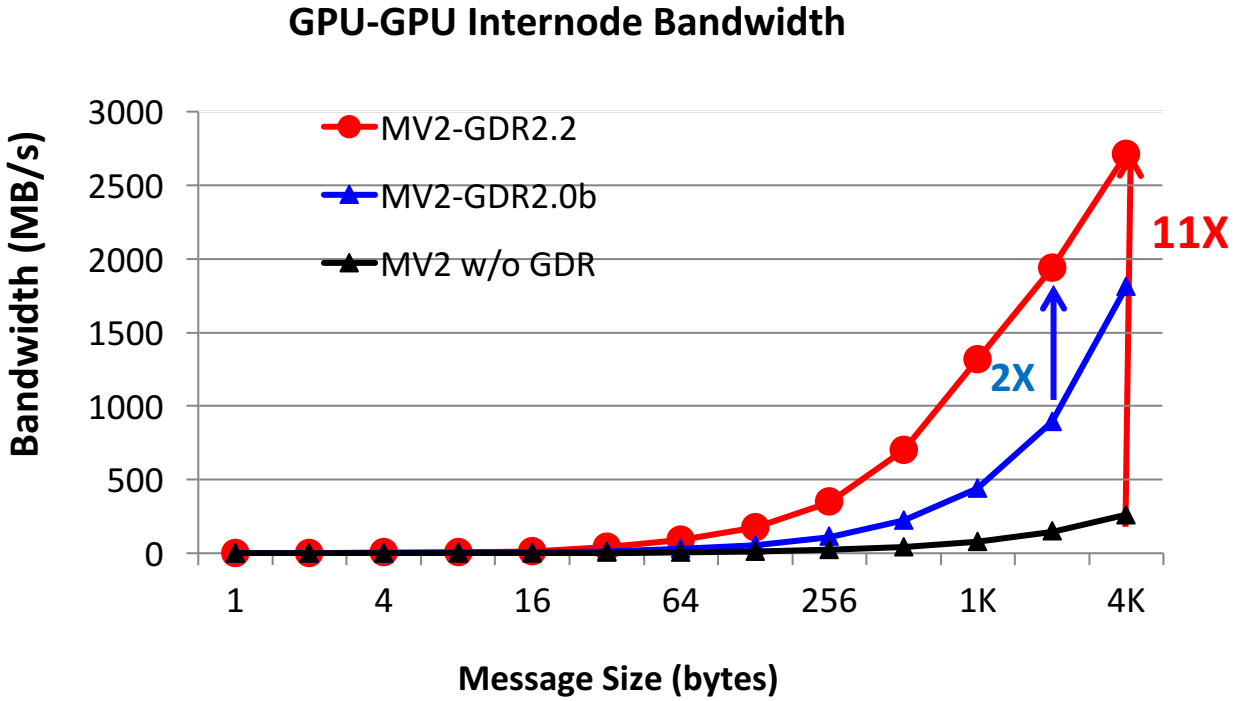
GPU Direct RDMA (GDR)

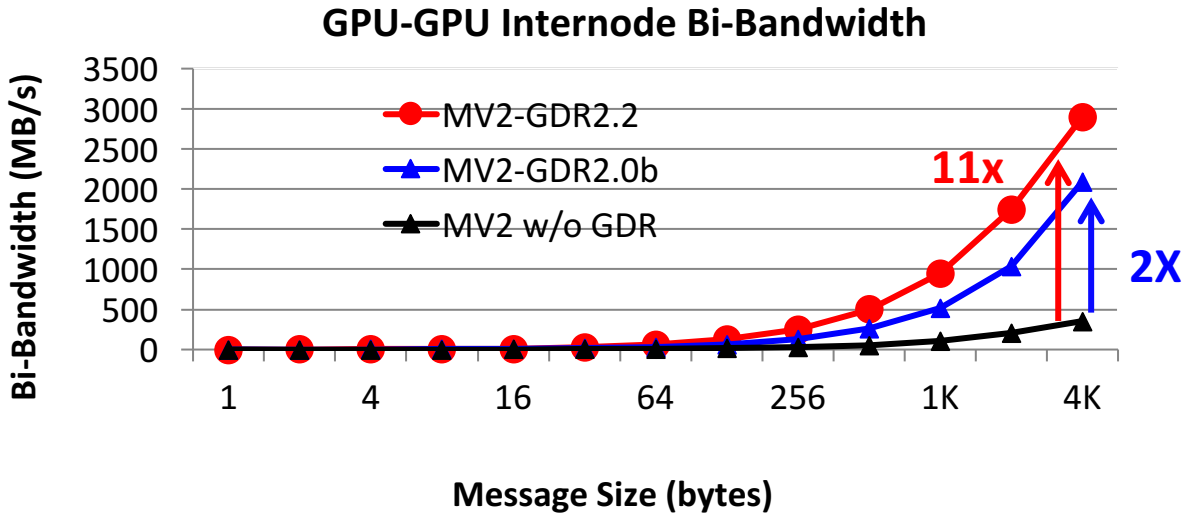
- **After CUDA 5.0, network adapter can directly read/write data from/to GPU device memory**
- Avoids copies through the host
- Fastest possible comm. between GPU and IB HCA
- Allows for better asynchronous communication
- CUDA-Aware MPI can take advantage of GDR to give very good performance



Performance of MVAPICH2-GPU with GPU-Direct RDMA (GDR)



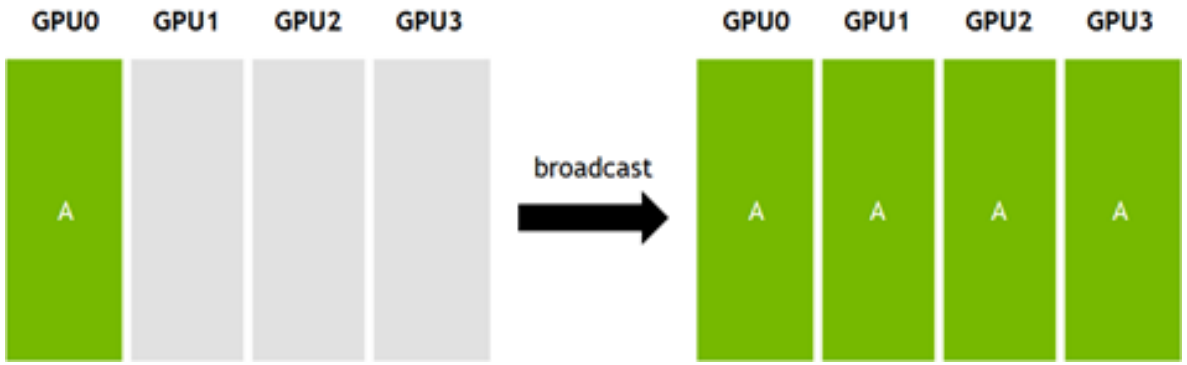


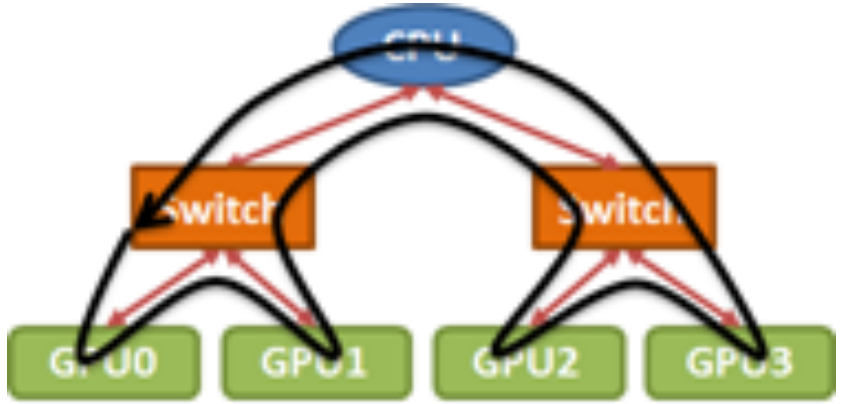


MVAPICH2-GDR-2.2
Intel Ivy Bridge (E5-2680 v2) node - 20 cores
NVIDIA Tesla K40c GPU
Mellanox Connect-X4 EDR HCA
CUDA 8.0
Mellanox OFED 3.0 with GPU-Direct-RDMA

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/

Gloo

- Gloo is a collective communications library by Facebook
  - https://github.com/facebookincubator/gloo
- It comes with a number of collective algorithms useful for machine learning applications
  - Barrier
  - Broadcast
  - Allreduce
- Transport of data between participating machines is abstracted so that IP can be used at all times, or InfiniBand (or RoCE) when available
- If InfiniBand transport is used, GPUDirect can be used to accelerate cross machine GPU-to-GPU memory transfers
- Implementation that works with system memory buffers, and one that works with NVIDIA GPU memory buffers. (CUDA-Aware)

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

Outline

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

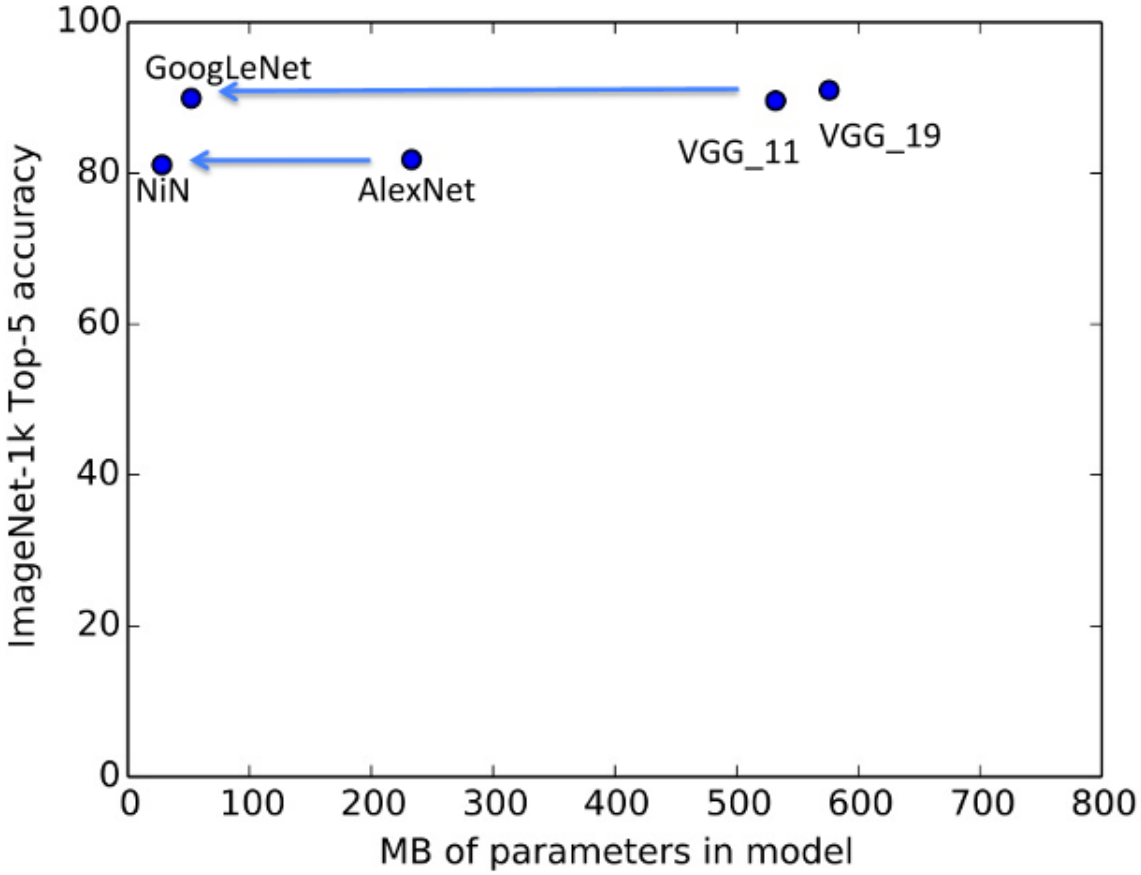
- What are the fundamental issues in designing DL frameworks?
  - Memory Requirements
  - Computation Requirements
  - Communication Overhead
- Why do we need to support distributed training?
  - To overcome the limits of single-node training
  - To better utilize hundreds of existing HPC Clusters
- What are the new design challenges brought forward by DL frameworks for Communication (MPI) runtimes?
  - Large Message Communication and Reductions
  - GPU Buffers
  - Device-specific Optimizations - GPU, KNL, TPU (in future)

Research Challenges: Co-Design Communication Runtimes

- Can GPU-optimized libraries like NCCL and Gloo provide scalable training for next-generation DNN architectures?
  - Limitations of Gloo and NCCL
  - Will work across nodes with newer designs (Case study #3, to be discussed later)
- What is the problem in Designing a DL Framework?
  - Isolated Designs at the application level may/can not provide the best performance
  - Primitives available but efficient support is missing (MPI_Iallreduce, MPI_Ibcast, etc.)
  - Expectation vs. Reality
- 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 runtime level?

Large Message Communication and Reduction

- What are the new requirements and expectations for Communication Runtimes?
  - Efficiently handle very-large buffer sizes
    - Megabytes (MB) for now
    - Expect Gigabytes (GB) in future
  - New algorithms and implementations will be needed!
  - GPU buffers in existing DL frameworks
  - Importance of efficient CUDA-Aware MPI will increase even more!



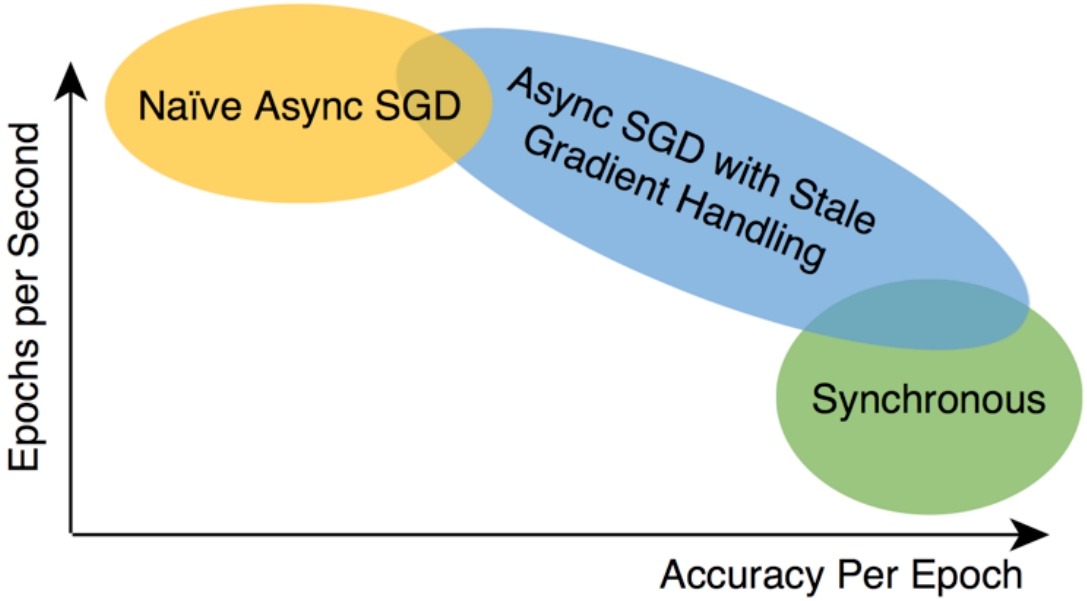
Courtesy: http://arxiv.org/abs/1511.00175

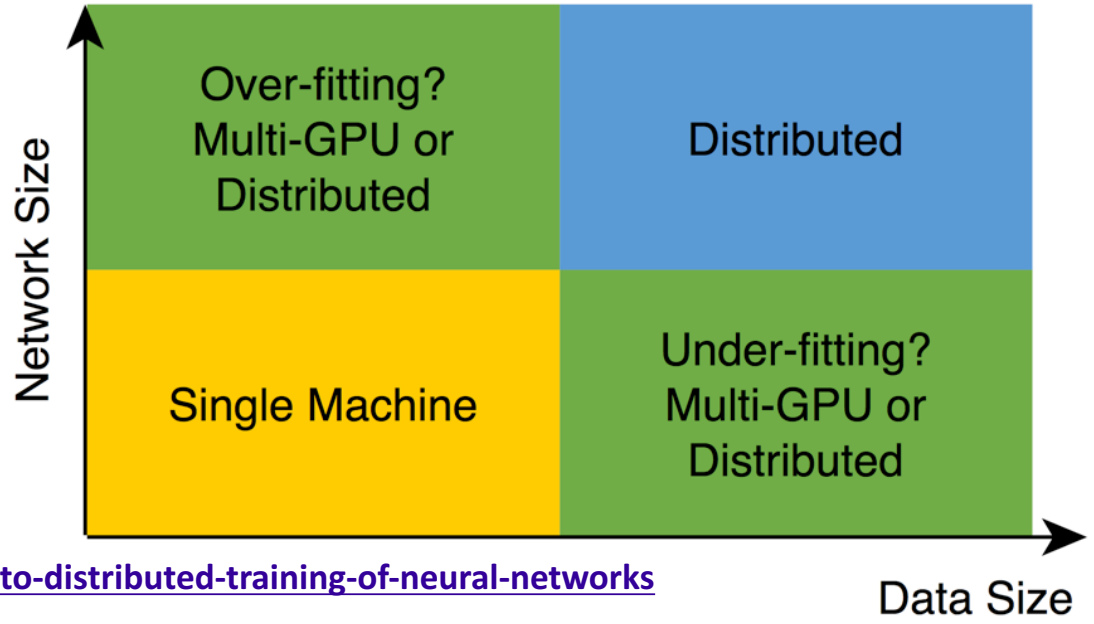
Efficient Collective Communication for Deep Learning

- Major MPI Collectives involved in Designing DL Frameworks
  - Large Message Broadcast – required for DNN parameter exchange
  - Large Message Reductions – needed for gradient accumulation from multiple solvers
  - Large Message Allreduce – use just one Allreduce instead of Reduce and Broadcast
- Are there libraries that can improve collective communication performance for GPU buffers?
  - NVIDIA NCCL 1.x single node only
  - NCCL 2.0 has support for multi-node communication
  - Facebook Gloo – requires Redis for Rendezvous
- Can MPI provide a holistic solution to this problem?

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

Outline

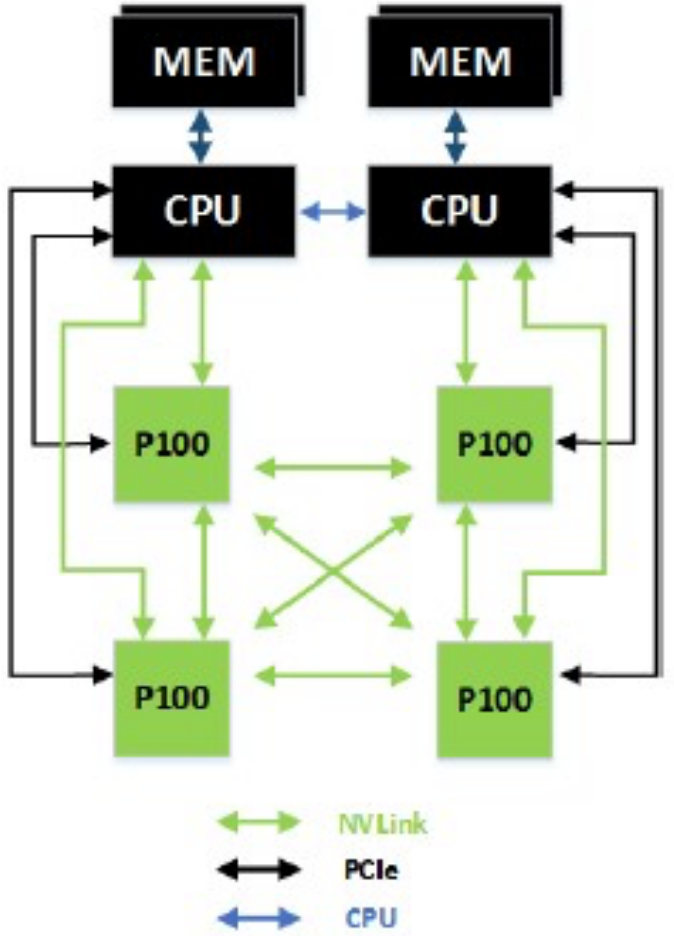
- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- **Case Studies and Demos**
- Open Issues and Challenges
- Conclusion

Case Studies and Demos: Exploiting HPC for DL

- **NVIDIA NCCL**
- Baidu-allreduce (+Demo)
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
  - MPI+NCCL for CUDA-Aware CNTK
  - OSU-Caffe (+Demo)
- MATEX TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs (+Demo)
- 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

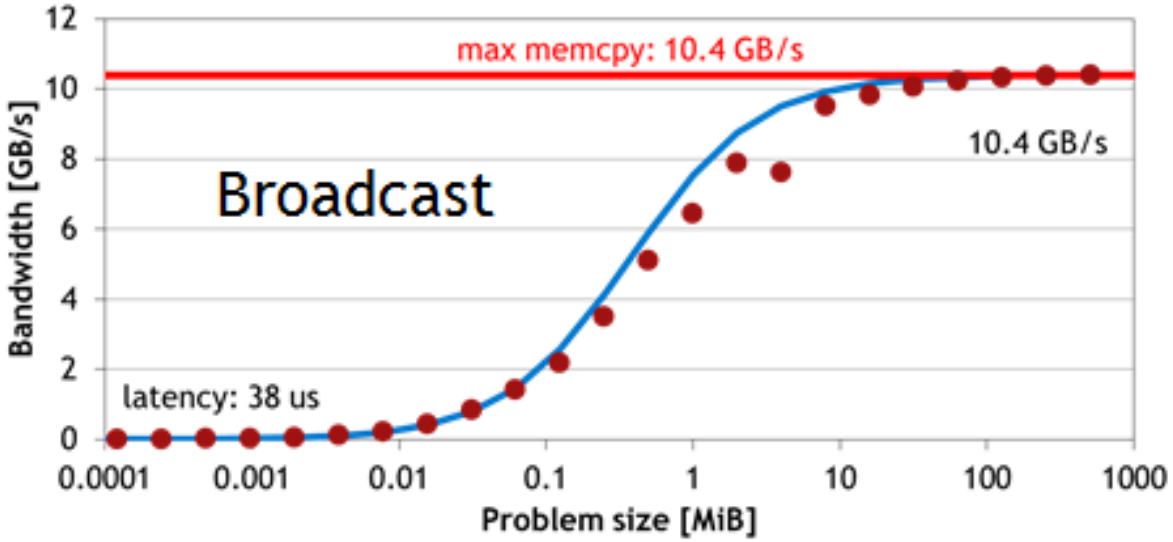
40 GB/s per GPU bidirectional to CPU

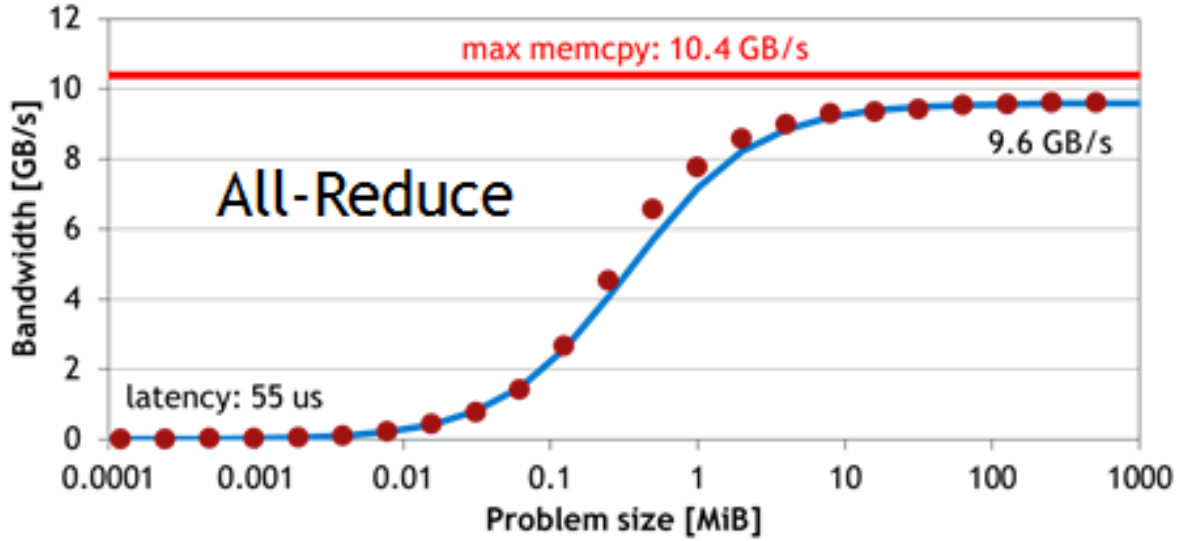
Direct Load/store access to CPU Memory

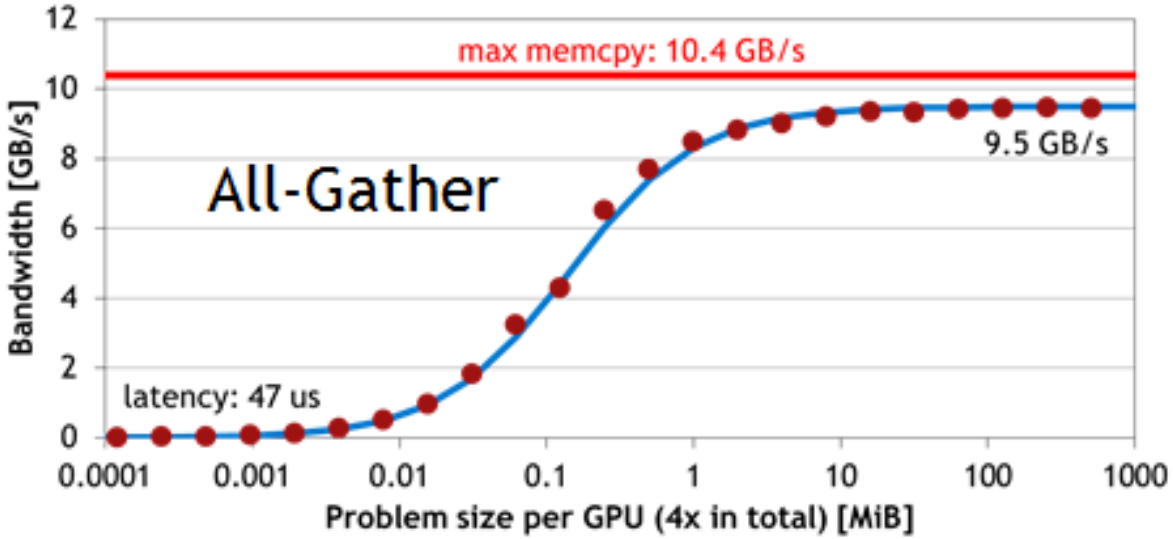
High Speed Copy Engines for bulk data movement

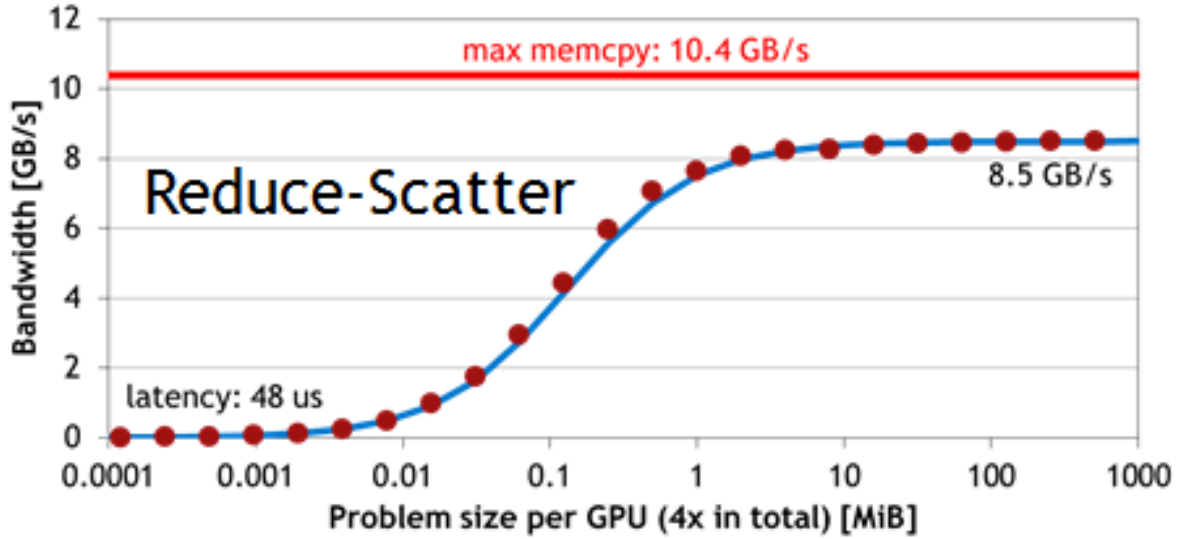
Courtesy: https://www.nextplatform.com/2016/05/04/nvlink-takes-gpu-acceleration-next-level/

Performance of NCCL 1.x Collectives



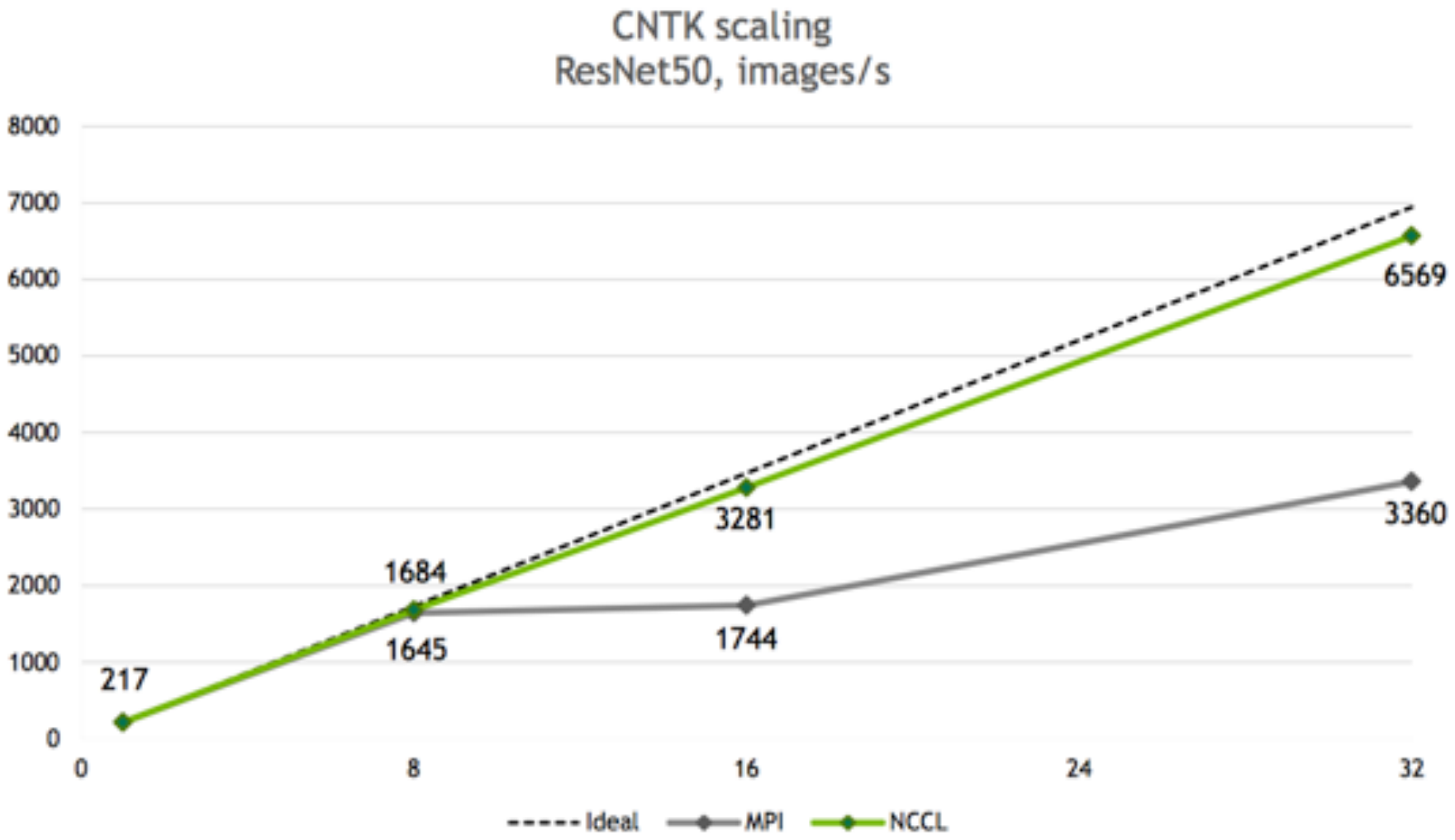






Courtesy: https://devblogs.nvidia.com/parallelforall/fast-multi-gpu-collectives-nccl/

NCCL 2: Multi-node GPU Collectives



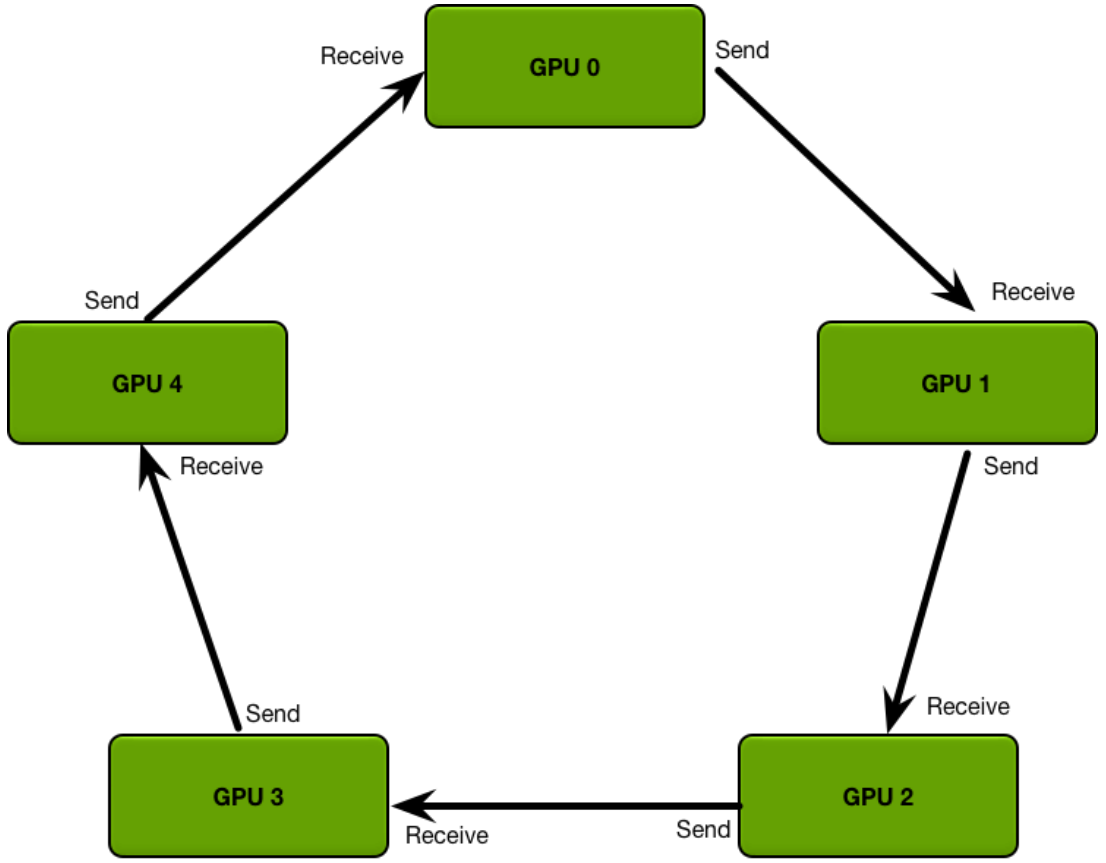
Courtesy: http://on-demand.gputechconf.com/gtc/2017/presentation/s7155-jeaugey-nccl.pdf

Case Studies and Demos: Exploiting HPC for DL

- NVIDIA NCCL
- **Baidu-allreduce (+Demo)**
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
  - MPI+NCCL for CUDA-Aware CNTK
  - OSU-Caffe (+Demo)
- MATEX TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs (+Demo)
- PowerAI DDL

Baidu-allreduce

- Baidu uses large message Allreduce collectives
- Performance degradation due to slow Allreduce in OpenMPI
- Proposed Solution:
  - Implement a Ring-Allreduce algorithm on top of point to point MPI primitives (Send/Recv) at the application level
- 2.5-3X better performance than OpenMPI Allreduce
- Used in the Deep Speech 2 paper*



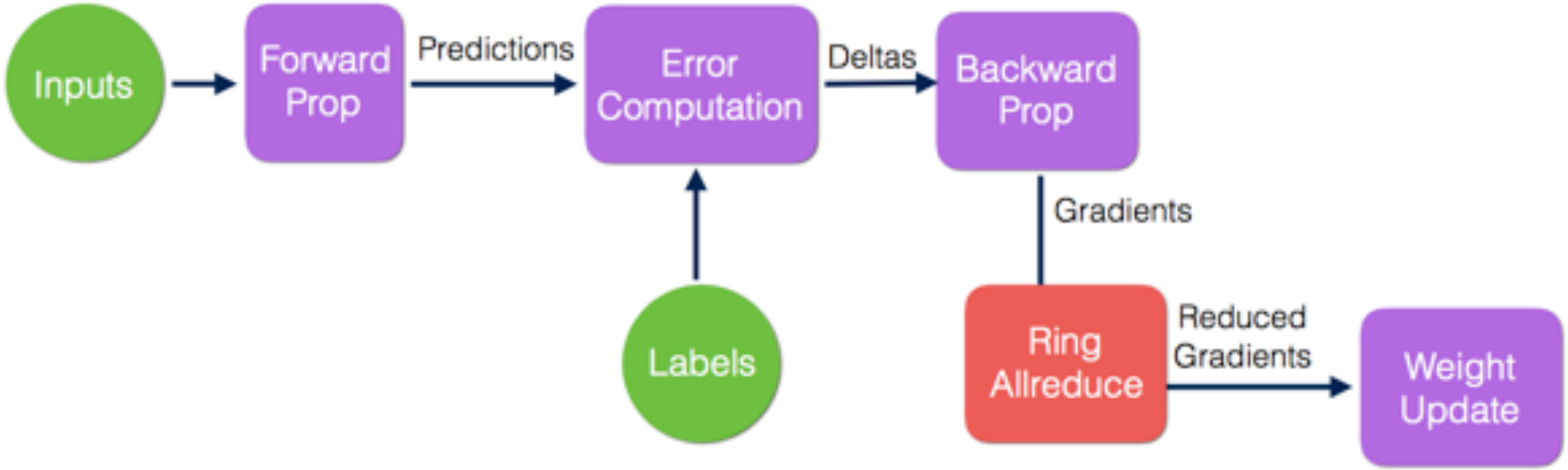
Courtesy: http://research.baidu.com/bringing-hpc-techniques-deep-learning/

*Amodei, Dario et al. "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin." *ICML* (2016).

Ring-Allreduce in TensorFlow

Scaling with TensorFlow

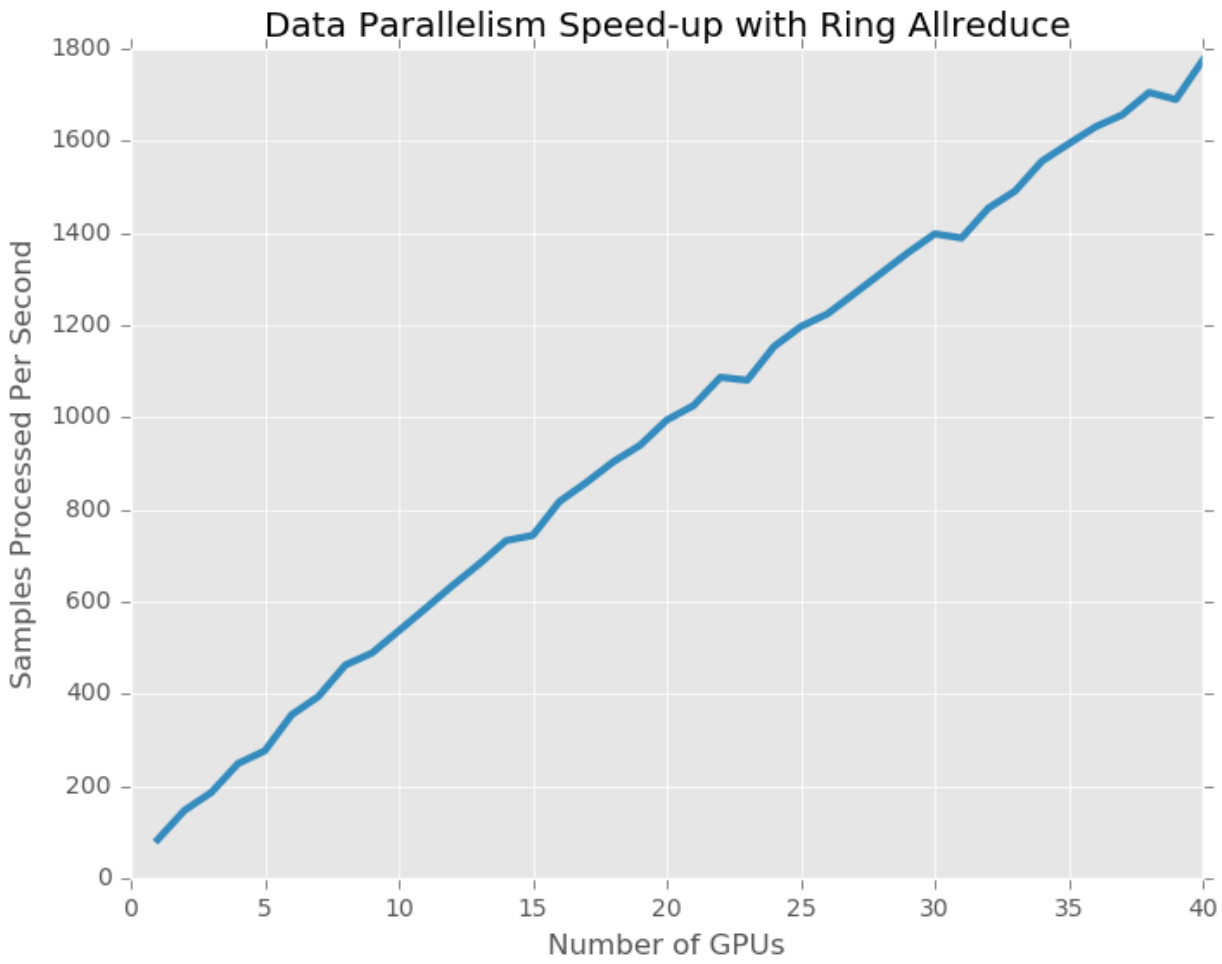
- Run many independent TensorFlow processes
- Insert allreduce as a node in the graph:



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

Data Parallel Training with Baidu-allreduce

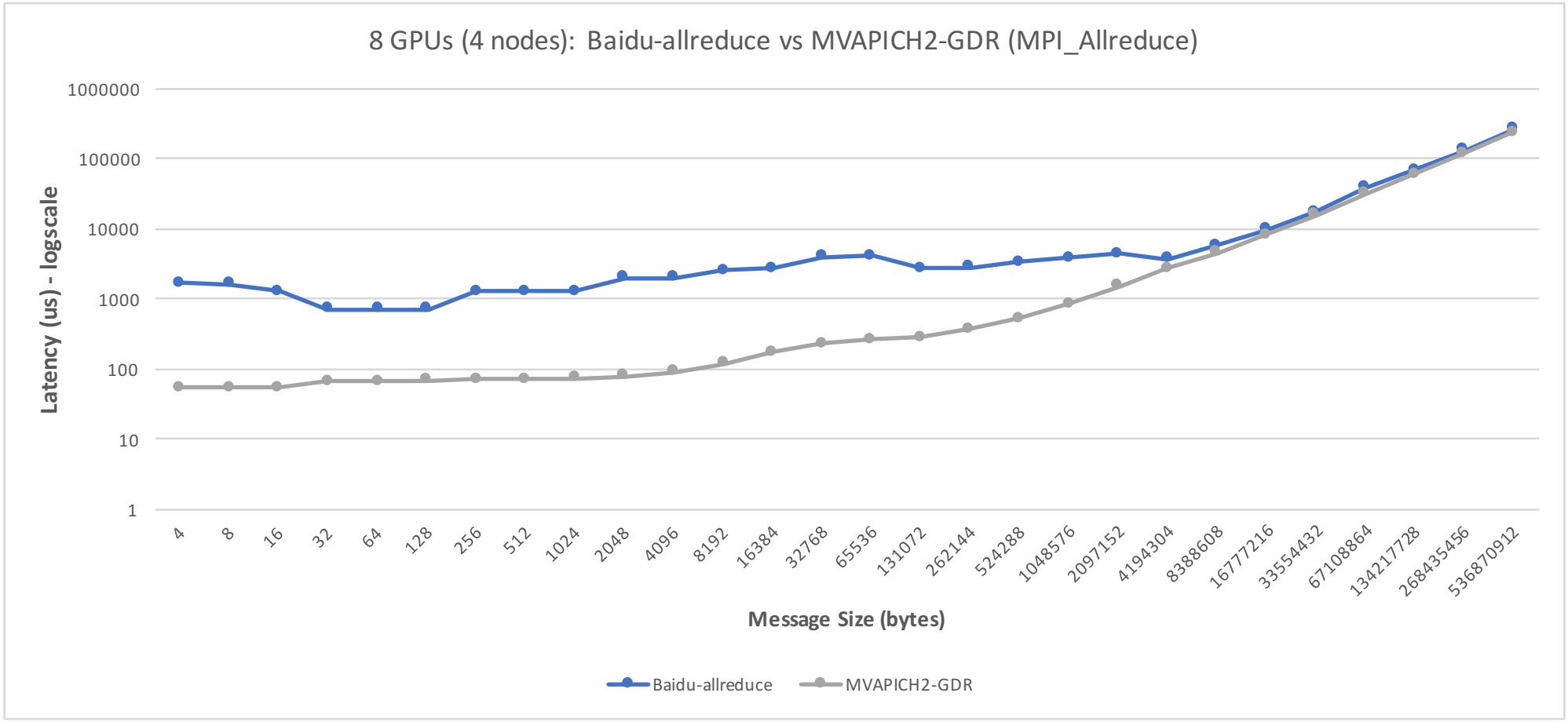
- Near-linear speedup for DNN training throughput (samples/second)
- The Allreduce design has been integrated in a TensorFlow contribution
- Details of the design are available from the Github site: https://github.com/baidu-research/tensorflow-allreduce



Courtesy: http://research.baidu.com/bringing-hpc-techniques-deep-learning/

MVAPICH2-GDR vs. Baidu-allreduce

- Initial Evaluation shows promising performance gains for MVAPICH2-GDR 2.3a* compared to Baidu-allreduce



**MVAPICH2-GDR 2.3a will be available soon*

Demo: Running Baidu-allreduce

1. Clone the repo

git clone https://github.com/baidu-research/baidu-allreduce

2. Setup MVAPICH2-GDR (userguide: http://mvapich.cse.ohio-state.edu/userguide/gdr/2.2)

export MV2HOME=/home/awan/hoti17-demo/mpi-builds/gdr2.3a-pre/install

export PATH=/home/awan/hoti17-demo/mpi-builds/gdr2.3a-pre/install/bin:$PATH

export LD_LIBRARY_PATH=/home/awan/hoti17-demo/mpi-builds/gdr2.3a-pre/install/lib:$LD_LIBRARY_PATH

3. Compile the allreduce tests using the included Makefile

make MPI_ROOT=/home/awan/hoti17-demo/mpi-builds/gdr2.3a-pre/install/ CUDA_ROOT=/usr/local/cuda

Baidu-allreduce Dependencies and Setup

- Baidu-allreduce is a library that implements an algorithm to perform Allreduce
- It depends on the underlying MPI implementation
- For this demo, we will use MVAPICH2-GDR 2.3a for building and running the tests
- **Example: Running Baidu-allreduce on 4 GPUs**

```
mpirun_rsh -np 4 -hostfile hosts \
MV2_USE_CUDA=1 \
MV2_GPUDIRECT_GDRCOPY_LIB=/opt/gdrcopy8.0/libgdrapi.so \
./allreduce-test gpu
```

Running MPI_Allreduce Benchmark

- **Example: Running MPI_Allreduce using MVAPICH2-GDR 2.3a on 4 GPUs**

```
mpirun_rsh -np 4 -hostfile hosts \
MV2_USE_CUDA=1 \
MV2_CUDA_USE_NAIVE=0 \
MV2_INTER_ALLREDUCE_TUNING=2 \
MV2_GPUDIRECT_GDRCOPY_LIB=/opt/gdrcopy8.0/libgdrapi.so \
$MV2HOME/libexec/osu-micro-benchmarks/get_local_rank \
$MV2HOME/libexec/osu-micro-benchmarks/mpi/collective/osu_allreduce -d cuda
-m 536870912
```

Performance Comparison: MVAPICH2-GDR vs. Baidu-allreduce

- For small and medium buffer sizes, MVAPICH2-GDR beats Baidu-allreduce significantly (**up to 20X better**)
- For large buffer sizes, MVAPICH2-GDR still outperforms Baidu-allreduce (**up to 2X better for 64 MB**)

Size (bytes)	Latency (us)
4	39.42
8	32.44
16	54.20
32	38.13
64	41.28
128	43.10
256	43.49
512	44.58
1024	46.84
2048	51.99
4096	63.09
8192	85.93
16384	135.71
32768	161.97
65536	141.42
131072	163.04
262144	204.31
524288	276.51
1048576	433.09
2097152	739.06
4194304	1264.40
8388608	2290.50
16777216	4269.73
33554432	8319.41
67108864	16360.03
134217728	31938.31
268435456	63265.43
536870912	125938.92

MVAPICH2-GDR

Size (bytes.	Latency (us)
4	829.64
8	817.326
16	582.546
32	570.465
64	567.269
128	1108.4
256	1110.64
512	1104.41
1024	1102.5
2048	1901.99
4096	1970.62
8192	1847.08
16384	2510
32768	2501
65536	1603.22
131072	1578.56
262144	1769.56
524288	1956.01
1048576	2397.11
2097152	2123.98
4194304	3172.35
8388608	5248.69
16777216	9332.51
33554432	17599.3
67108864	33537.1
134217728	62901.5
268435456	115821
536870912	186610

Baidu-allreduce

Baidu-allreduce Demo: Steps to execute..

1. Go to the baidu-allreduce directory

cd /home/awan/projects/baidu-allreduce

2. Allocate nodes and setup hostnames in file: hosts

3. Run Baidu-allreduce on 2 GPUs

sh run-baidu.sh 2

4. Run MVAPICH2-GDR osu_allreduce benchmark on 2 GPUs

sh run-mv2.sh 2

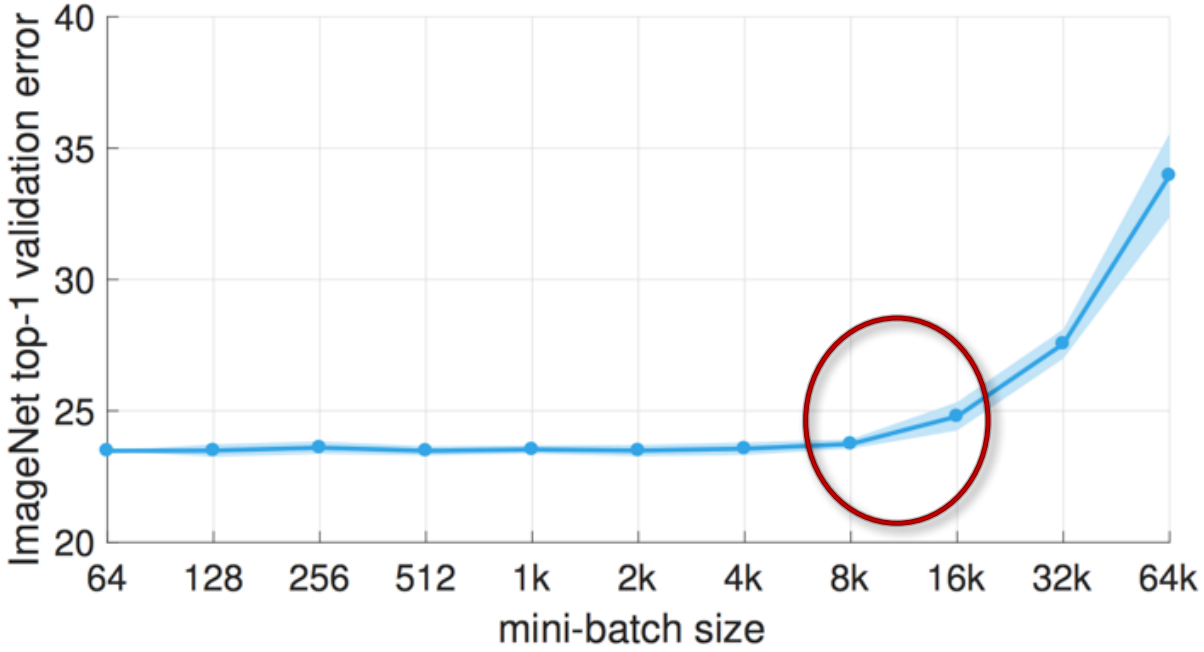
Case Studies and Demos: Exploiting HPC for DL

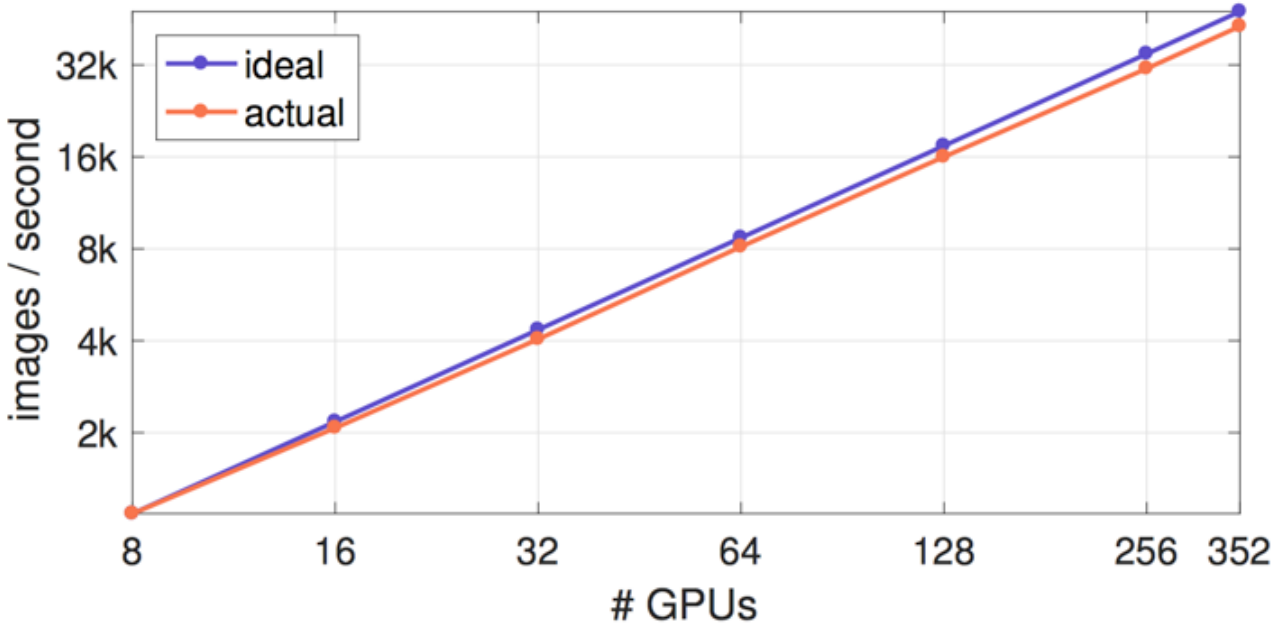
- NVIDIA NCCL
- Baidu-allreduce (+Demo)
- **Facebook Gloo**
- Co-design MPI runtimes and DL Frameworks
  - MPI+NCCL for CUDA-Aware CNTK
  - OSU-Caffe (+Demo)
- MATEX TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs (+Demo)
- PowerAI DDL

Facebook Gloo and Caffe2

- Gloo: Recently introduced by Facebook for Collective communication on GPUs
- 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.***

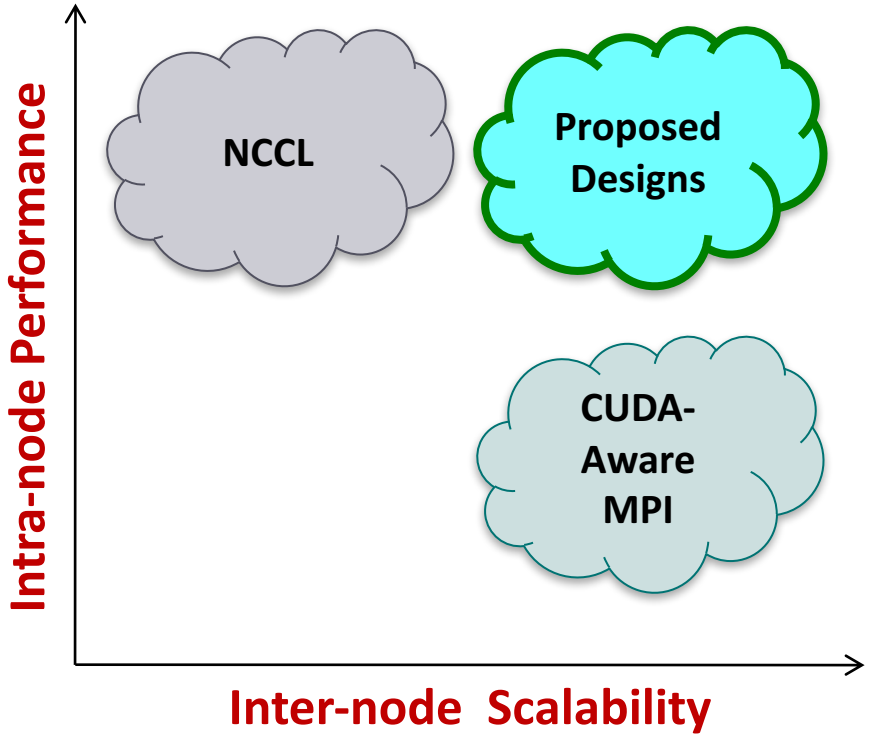
Courtesy: https://research.fb.com/publications/imagenet1kin1h/

Case Studies and Demos: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce (+Demo)
- Facebook Gloo
- **Co-design MPI runtimes and DL Frameworks**
  - **MPI+NCCL for CUDA-Aware CNTK**
  - OSU-Caffe (+Demo)
- MATEX TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs (+Demo)
- PowerAI DDL

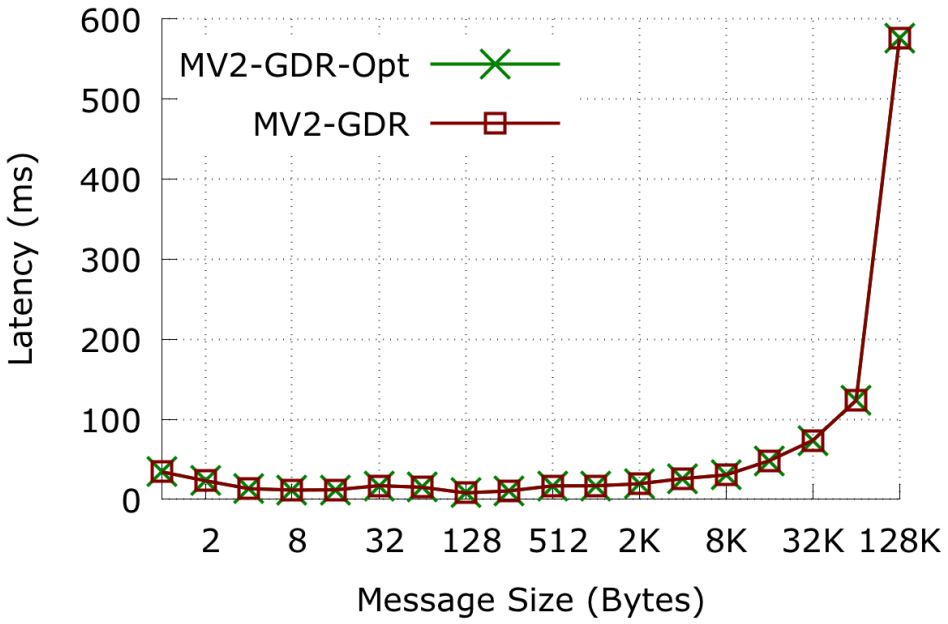
MPI+NCCL: Can we exploit NCCL to accelerate MPI?

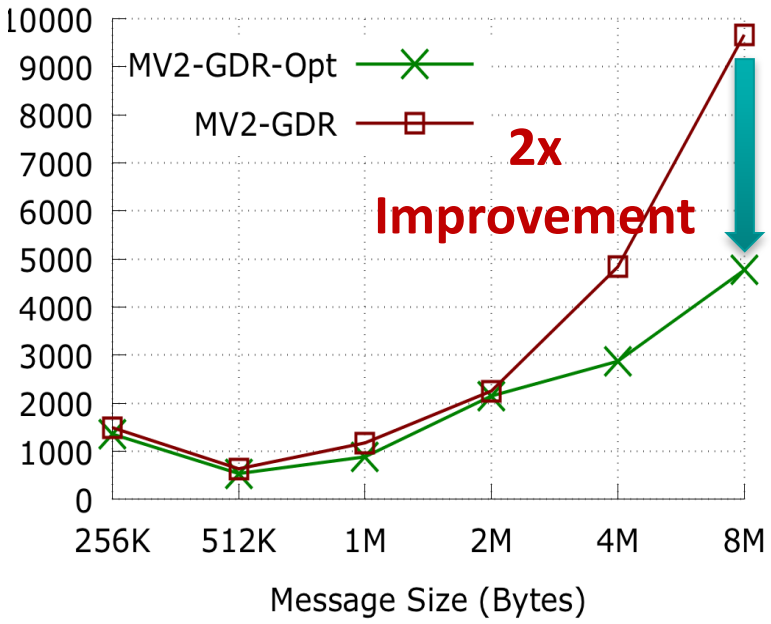
- CUDA-Aware MPI provides excellent performance for small and medium message sizes
- NCCL has overhead for small messages but provides excellent performance for large messages
- Can we have designs that provide good performance for intra-node communication and inter-node scalability?
  - Exploit NCCL for intra-node inter-GPU communication
  - Design and utilize existing Inter-node communication in MVAPICH2-GDR

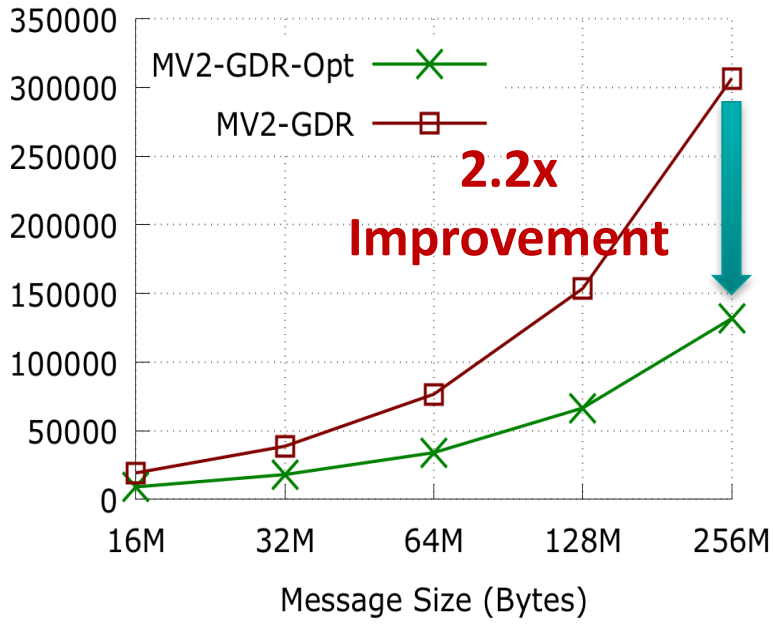


A. A. Awan, K. Hamidouche, A. Venkatesh, and D. K. Panda, Efficient Large Message Broadcast using NCCL and CUDA-Aware MPI for Deep Learning. In *Proceedings of the 23rd European MPI Users' Group Meeting* (EuroMPI 2016). [Best Paper Nominee]

Performance Comparison of Proposed vs. Existing Designs



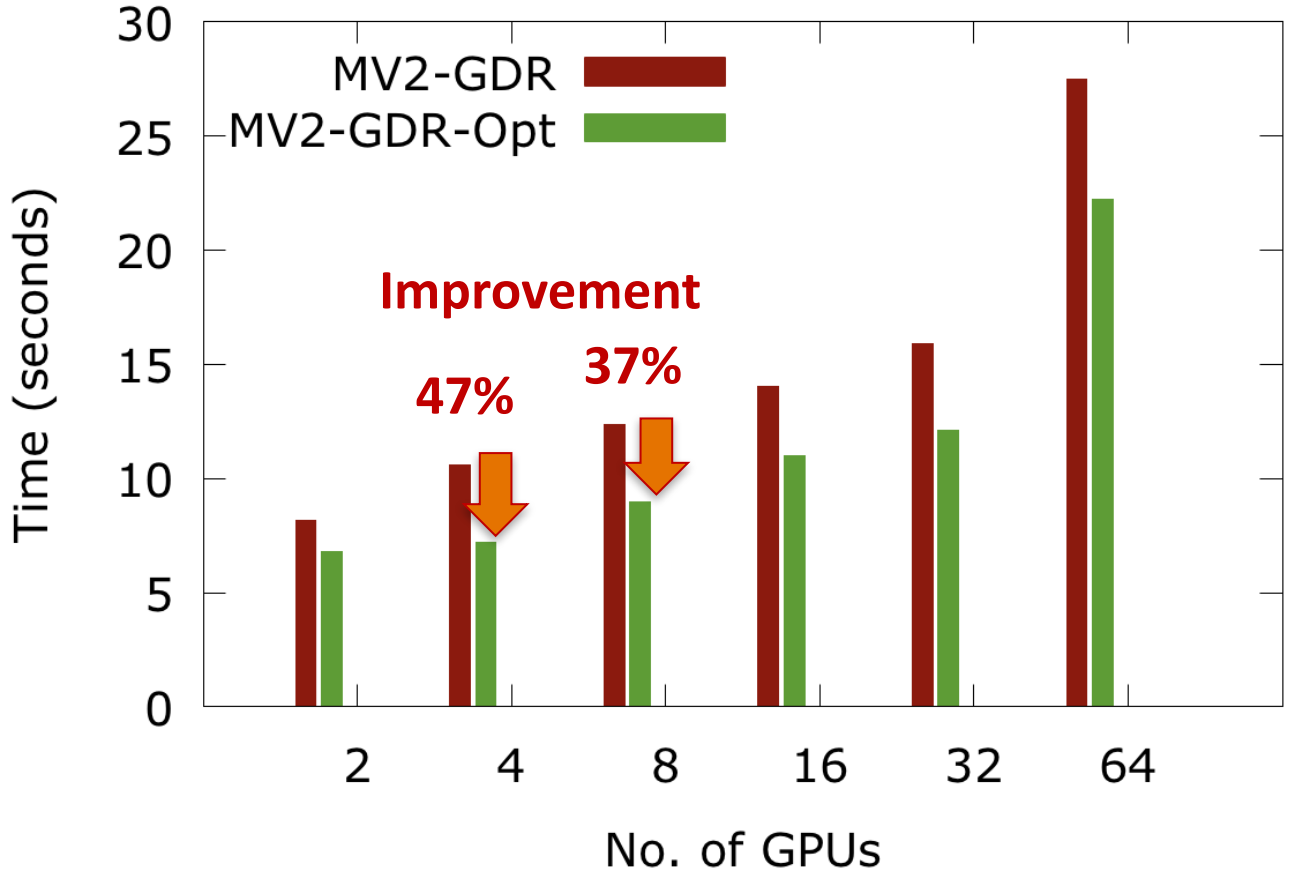




- MV2-GDR-Opt performs as good as MV2-GDR for small and medium messages (up to 2M)
- For large messages, MV2-GDR-Opt provides up to **2.2x improvement** over MV2-GDR

Application Performance with Microsoft CNTK (64 GPUs)

- Microsoft CNTK is a popular and efficient DL framework
- CA-CNTK is a CUDA-Aware version developed at OSU
- Proposed Broadcast provides up to **47%** improvement in Training time for the **VGG** network

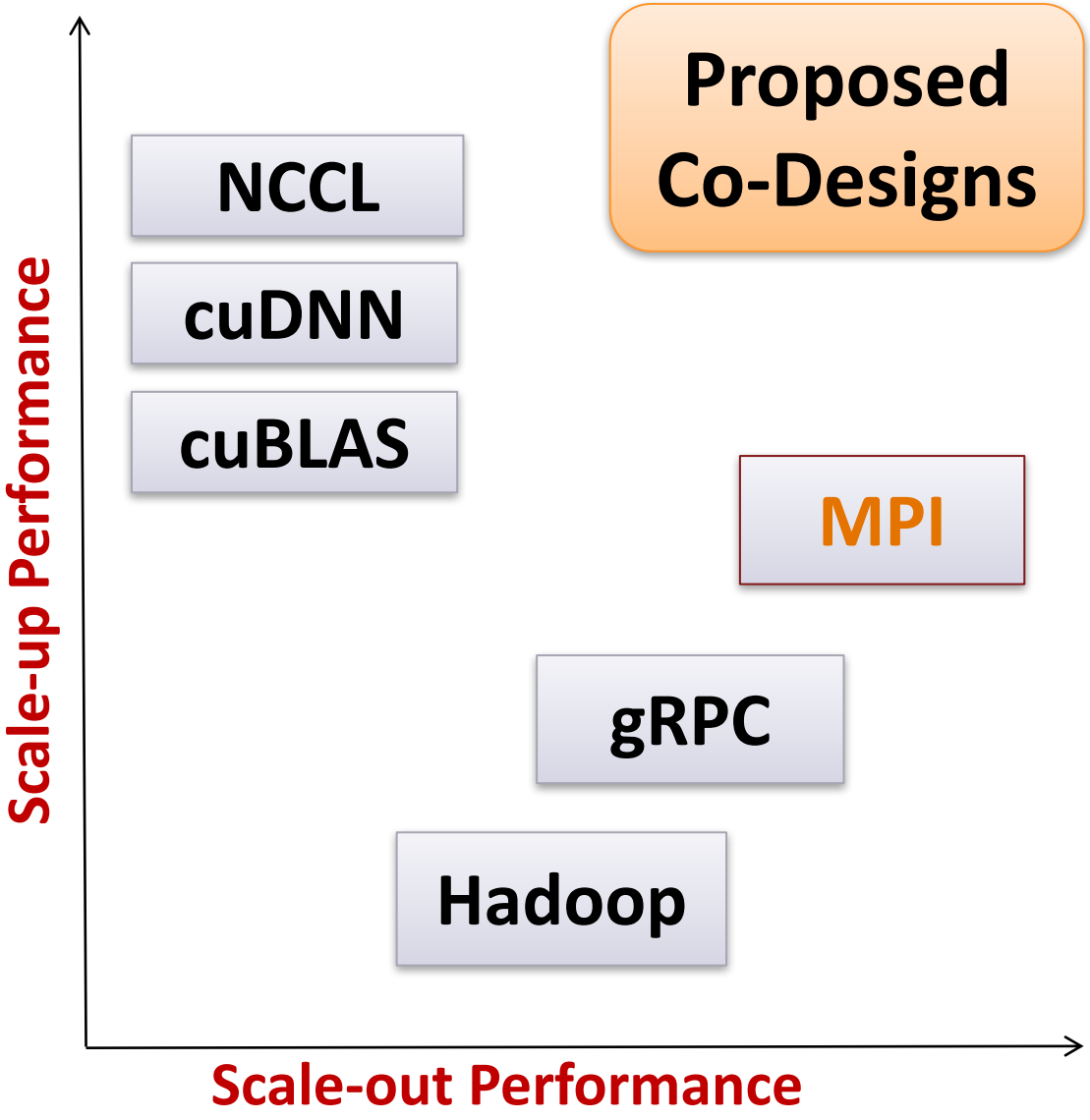


Case Studies and Demos: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce (+Demo)
- Facebook Gloo
- **Co-design MPI runtimes and DL Frameworks**
  - MPI+NCCL for CUDA-Aware CNTK
  - **OSU-Caffe (+Demo)**
- MATEX TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs (+Demo)
- PowerAI DDL

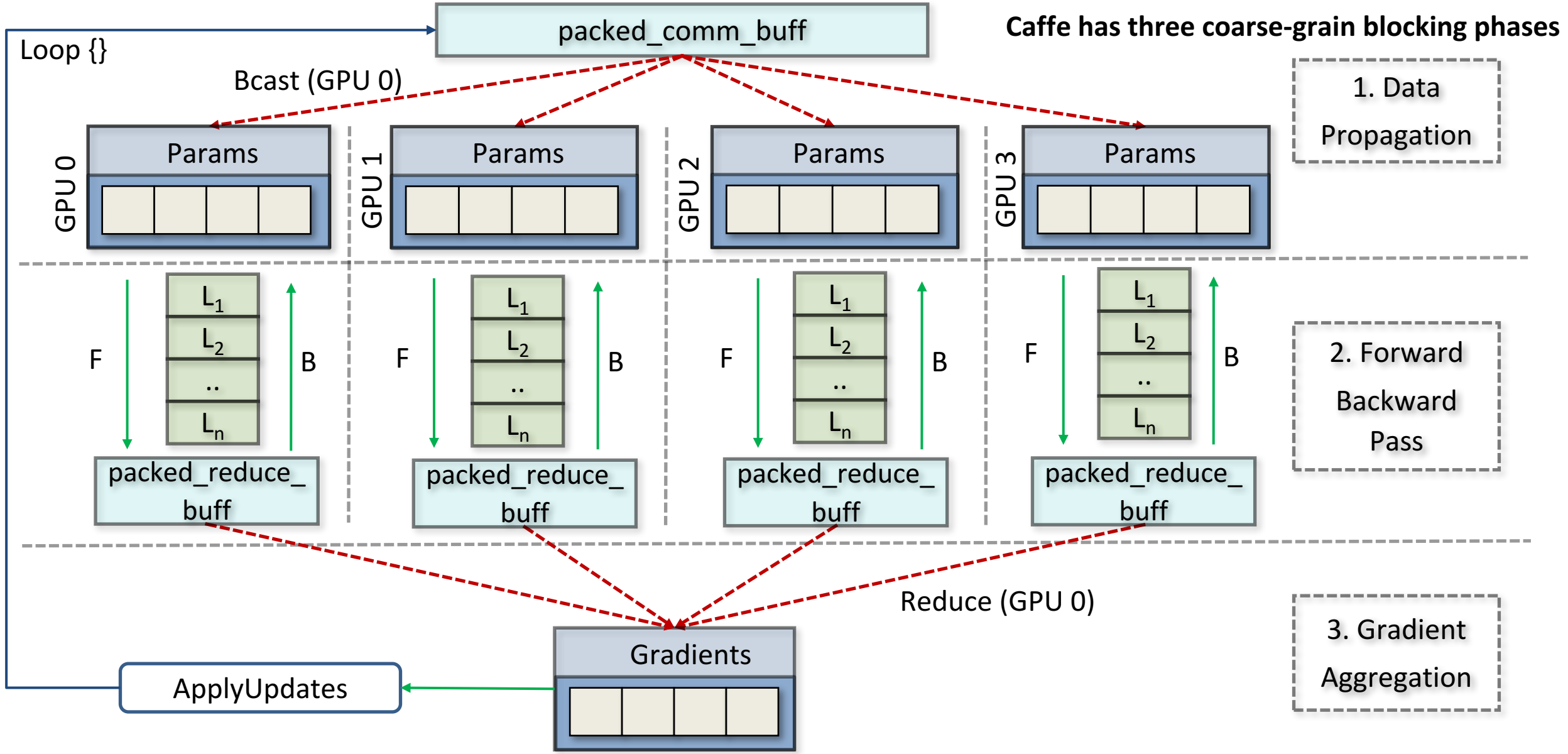
Can we co-design MPI and Caffe?

- Existing State-of-the-art
  - cuDNN, cuBLAS, NCCL --> **scale-up** performance
  - CUDA-Aware MPI --> **scale-out** performance
    - For small and medium message sizes only!
- Proposed: Can we **co-design** the MPI runtime (**MVAPICH2-GDR**) and the DL framework (**Caffe**) to achieve both?
  - Efficient **Overlap** of Computation and Communication
  - Efficient **Large-Message** Communication (Reductions)
  - What **application co-designs** are needed to exploit **communication-runtime co-designs**?



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)

Limitations of Existing Designs in Caffe

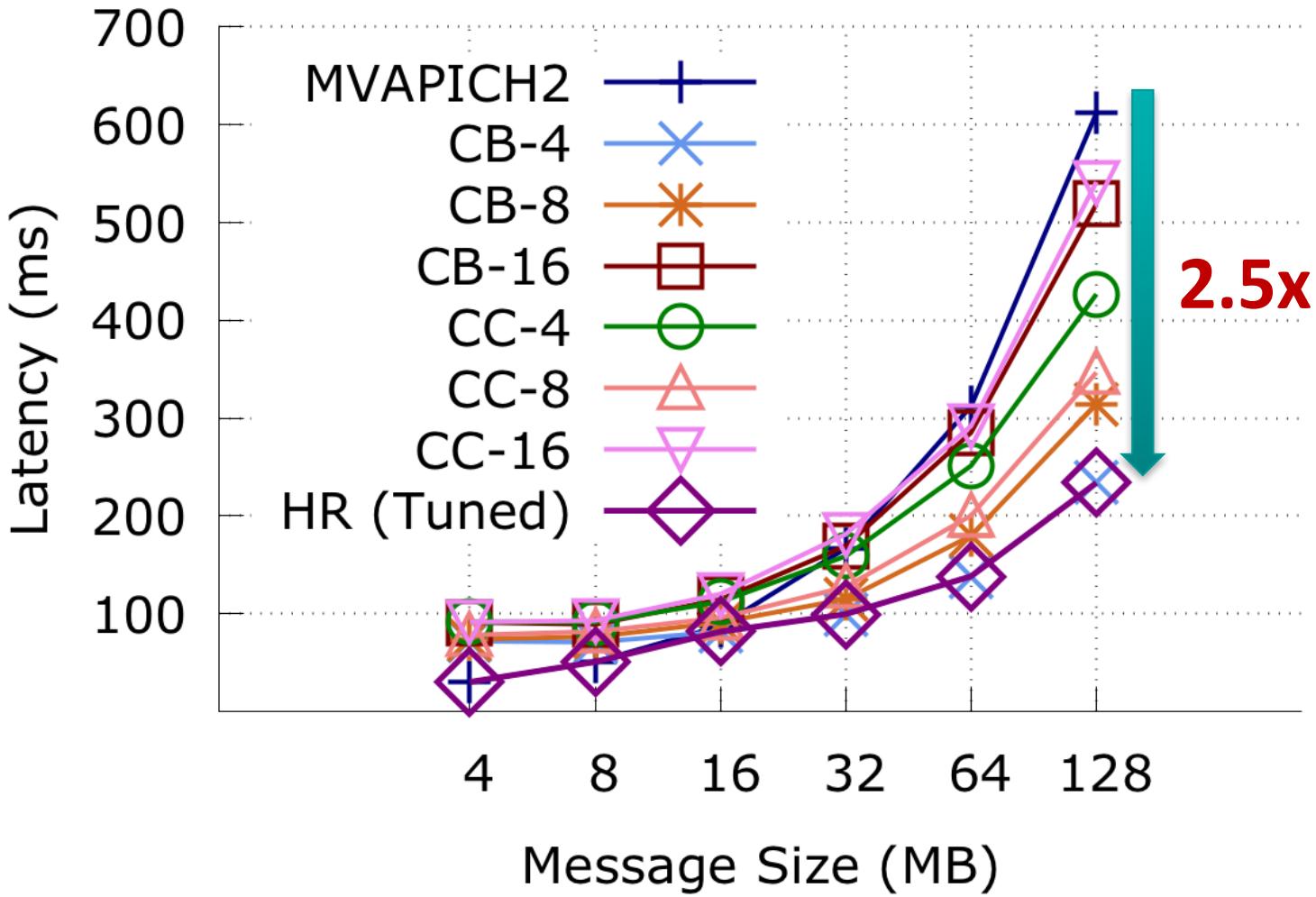


OSU-Caffe: Proposed Co-Design Overview

- 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)

Hierarchical Reduce (HR) - 160 GPUs

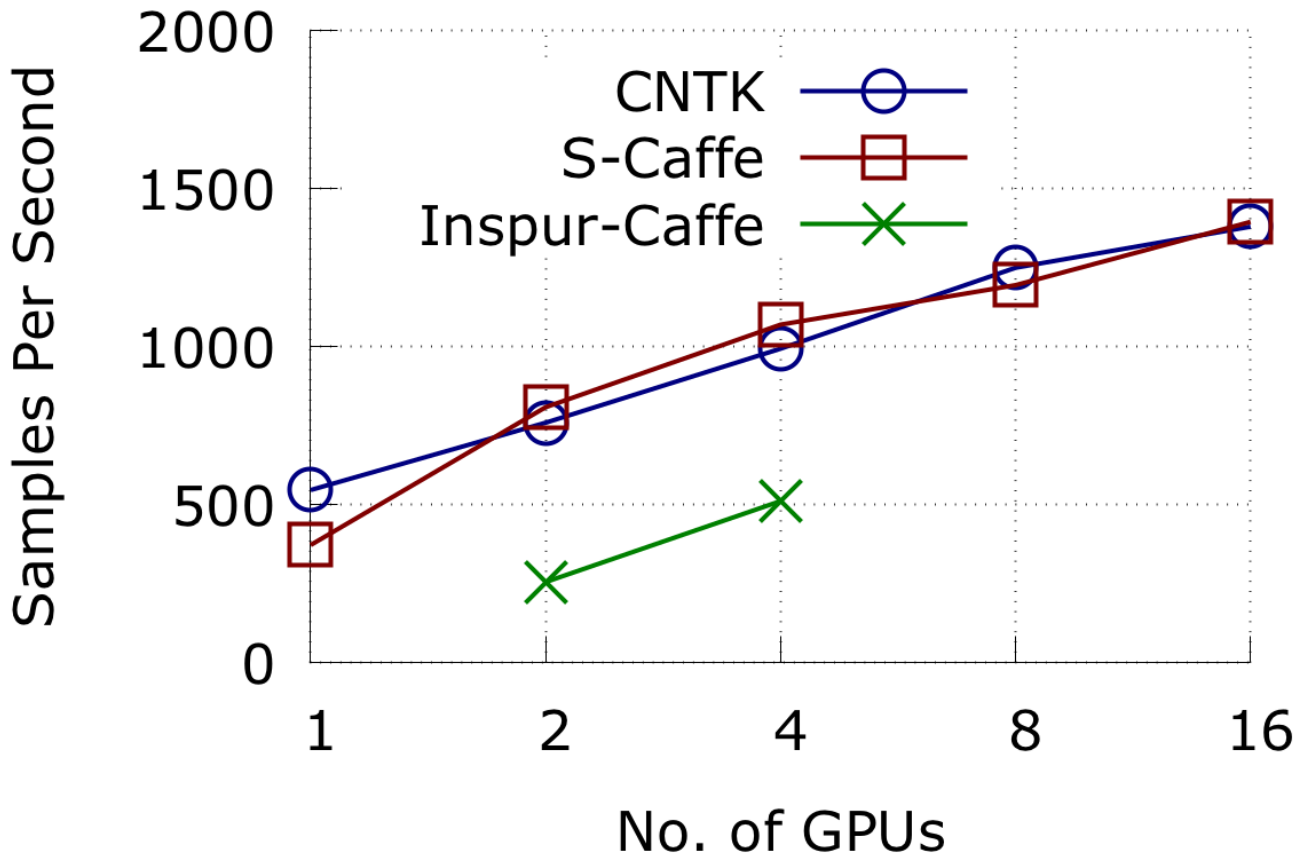


- Various designs to achieve best performance across platforms
- Proposed HR-tuned provides the best performance (up to 2.5X)

Comparison (AlexNet): S-Caffe, Inspur-Caffe, and CNTK

- AlexNet – the benchmark network for Image recognition challenges

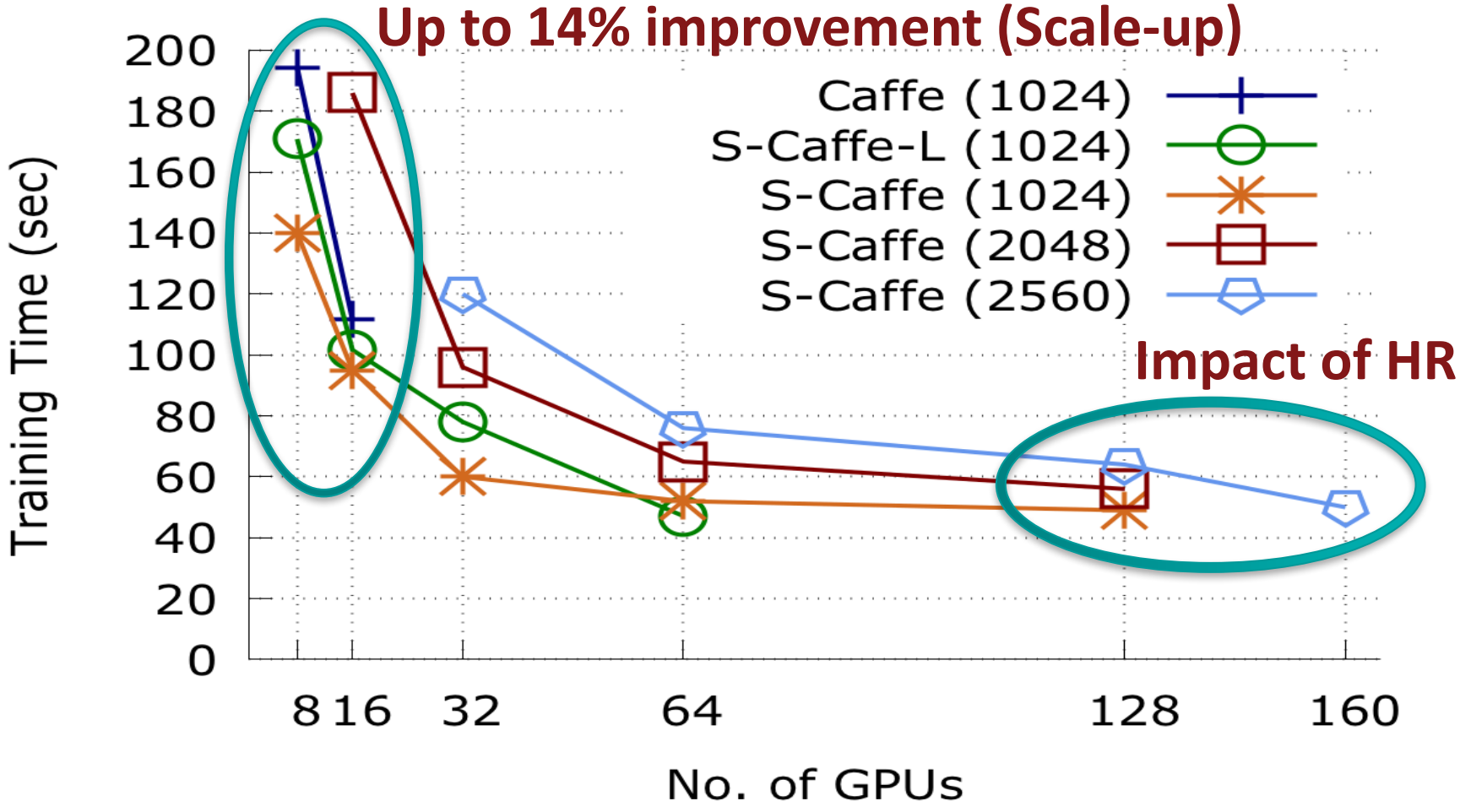
- Notoriously hard to scale-out on multiple nodes
- Main reason: increasing communication overhead!
- Large number of parameters ~ 64 Million
- Buffer Size ~ 256 MB



S-Caffe delivers better or comparable performance with other multi-node capable DL frameworks

S-Caffe: GoogLeNet (160 GPUs)

- GoogLeNet is another popular DL network
- 13 million parameters
- Comm. Buffer > 50 MB



- S-Caffe provides scale-out to 160 GPUs

Demo: OSU-Caffe

- Download and setup OSU-Caffe
- OSU-Caffe uses the same build/tools/caffe binary with some modifications
- ImageNet training will require a separate download and setup. Please consult default Caffe setup
- Auxiliary dataset can be downloaded using ‘get_ilsvrc_aux.sh’ script

```
#Download and Setup
wget http://hidl.cse.ohio-state.edu/download/hidl/osu-caffe/0.9/mofed3.2/osu-caffe-0.9-cuda7.5-gnu-el
7.centos.tgz
tar -xf osu-caffe-0.9-cuda7.5-gnu-el7.centos.tgz
cd osu-caffe-0.9-cuda7.5-gnu-el7.centos
./install-nonroot.sh

#Run command
./opt/osu-caffe/osu-caffe-gnu/0.9/build/tools/caffe

#Get dataset extras
cd ./opt/osu-caffe/osu-caffe-gnu/0.9/
./data/ilsvrc12/get_ilsvrc_aux.sh
```

Setting up OSU-Caffe and dependencies

```
export CAFROOT=/home/awan/hoti17-demo/osu-caffe/osu-caffe-0.9-cuda7.5-gnu-el7.centos/opt/osu-caffe
export LD_LIBRARY_PATH=$CAFROOT/boost/1_58_0/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$CAFROOT/lmdb/0.9.17/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$CAFROOT/leveldb/1.18/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$CAFROOT/protobuf/2.6.1/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$CAFROOT/gflags/2.1.2/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$CAFROOT/glog/0.3.4/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$CAFROOT/hdf5/1.8.16/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$CAFROOT/opencv/3.1.0/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$CAFROOT/snappy/1.1.3/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/home/awan/downloads/cudnn-v5/cuda/lib64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/opt/intel/parallel_studio_xe_2017_update1/mkl/lib/intel64:$LD_LIBRARY_PATH

#Add caffe to PATH
export PATH=$CAFROOT/osu-caffe-gnu/0.9/build/tools/:$PATH

#setup MPI
export PATH=/home/awan/hoti17-demo/mpi-builds/gdr2.2/install/bin:$PATH
export LD_LIBRARY_PATH=/home/awan/hoti17-demo/mpi-builds/gdr2.2/install/lib:$LD_LIBRARY_PATH
export MV2_GPUDIRECT_GDRCOPY_LIB=/opt/gdrcopy8.0/libgdrapi.so
export MV2_USE_CUDA=1
```

Testing the Caffe binary included in OSU-Caffe

- Please note new options in OSU-Caffe
- -scal weak
- -scal strong (default)
- -amode 1,2, and 3
- -pmode 1,2, and 3
- By default, OSU-Caffe automatically selects them for you
  - Provided for advanced tuning for newer unknown models

```
[awan@gpu01 osu-caffe]$ caffe
caffe: command line brew
usage: caffe <command> <args>

commands:
  train           train or finetune a model
  test            score a model
  device_query    show GPU diagnostic information
  time            benchmark model execution time

  Flags from tools/caffe.cpp:
    -amode (Optional; select Optimization Level for Aggregation) type: int32
      default: 1
    -gpu (Optional; run in GPU mode on given device IDs separated by ','.Use
      '-gpu all' to run on all available GPUs. The effective training batch
      size is multiplied by the number of devices.) type: string default: ""
    -iterations (The number of iterations to run.) type: int32 default: 50
    -model (The model definition protocol buffer text file..) type: string
      default: ""
    -pmode (Optional; select Optimization Level for Propagation) type: int32
      default: 1
    -scal (Use -scal strong or weak; 'strong' means that the batch size
      specified in the prototxt file will be divided by the number of  solvers.
      'weak' is the scaling mode where effective batch size becomes batch_size
      * num_solvers) type: string default: "strong"
    -sighup_effect (Optional; action to take when a SIGHUP signal is received:
      snapshot, stop or none.) type: string default: "snapshot"
    -sigint_effect (Optional; action to take when a SIGINT signal is received:
      snapshot, stop or none.) type: string default: "stop"
    -snapshot (Optional; the snapshot solver state to resume training.)
      type: string default: ""
    -solver (The solver definition protocol buffer text file.) type: string
      default: ""
    -weights (Optional; the pretrained weights to initialize finetuning,
      separated by ','. Cannot be set simultaneously with snapshot.)
      type: string default: ""
```

Running OSU-Caffe to train AlexNet

- The run command can be used to run across 4 GPUs (gpu01 and gpu02 are hostnames)

```
[awan@gpu01 0.9]$ mpirun_rsh -export-all -np 4 gpu01 gpu01 gpu02 gpu02 MV2_USE_CUDA=1 caffe train -solver models/bvlc_reference_caffenet/solver.prototxt

I0827 09:07:25.065896 12915 solver.cpp:434] Solving CaffeNet
I0827 09:07:25.065898 12915 solver.cpp:435] Learning Rate Policy: step
I0827 09:07:25.562651 12915 solver.cpp:323] Iteration 0, loss = 7.4253
I0827 09:07:25.562703 12915 solver.cpp:341] Rank : 0    Train net output #0: loss = 7.4253 (* 1 = 7.4253 loss)
I0827 09:07:25.730669 12915 solver.cpp:758] Iteration 0, lr = 0.01
Could not create logging file: No such file or directory
COULD NOT CREATE A LOGGINGFILE 20170827-090731.12915!I0827 09:07:31.228024 12915 solver.cpp:323] Iteration 20, loss = 7.19989
I0827 09:07:31.228121 12915 solver.cpp:341] Rank : 0    Train net output #0: loss = 7.19989 (* 1 = 7.19989 loss)
I0827 09:07:31.297372 12915 solver.cpp:758] Iteration 20, lr = 0.01
I0827 09:07:36.685904 12915 solver.cpp:323] Iteration 40, loss = 6.91773
I0827 09:07:36.685963 12915 solver.cpp:341] Rank : 0    Train net output #0: loss = 6.91773 (* 1 = 6.91773 loss)
I0827 09:07:36.757009 12915 solver.cpp:758] Iteration 40, lr = 0.01
I0827 09:07:42.125658 12915 solver.cpp:323] Iteration 60, loss = 6.93448
I0827 09:07:42.125705 12915 solver.cpp:341] Rank : 0    Train net output #0: loss = 6.93448 (* 1 = 6.93448 loss)
I0827 09:07:42.196094 12915 solver.cpp:758] Iteration 60, lr = 0.01
I0827 09:07:47.545716 12915 solver.cpp:323] Iteration 80, loss = 6.89837
I0827 09:07:47.545763 12915 solver.cpp:341] Rank : 0    Train net output #0: loss = 6.89837 (* 1 = 6.89837 loss)
I0827 09:07:47.614868 12915 solver.cpp:758] Iteration 80, lr = 0.01
I0827 09:07:52.757889 12915 solver.cpp:636] Snapshotting to binary proto file models/bvlc_reference_caffenet/caffenet_train_it
I0827 09:07:54.012820 12915 solver.cpp:921] Snapshotting solver state to binary proto file models/bvlc_reference_caffenet/caff
olverstate
I0827 09:07:54.683089 12915 solver.cpp:478] Iteration 100, loss = 6.89293, rank = 0
I0827 09:07:54.683127 12915 solver.cpp:485] Optimization Done.
I0827 09:07:54.683151 12915 caffe.cpp:329] Optimization Done.
I0827 09:07:54.683341 12915 caffe.cpp:351]  Avg. Time Taken: 32.3766
```

Decreasing Loss as training proceeds

Accumulated Time Taken

Strong Scaling for AlexNet Training (BatchSize 128)

- Strong Scaling AlexNet with small batch-size is very hard!

- $ sh run-alexnet.sh 1

I0827 15:49:29.788367 4225 caffe.cpp:351] Avg. Time Taken: 35.2652

- $ sh run-alexnet.sh 2

I0827 15:50:19.963081 4298 caffe.cpp:351] Avg. Time Taken: 26.3524

- $ sh run-alexnet.sh 4

I0827 15:52:30.130556 4431 caffe.cpp:351] Avg. Time Taken: 26.0737

Larger Batch Size to scale better! (BatchSize 256)

- Better Scaling with a larger batch-size!

- $ sh run-alexnet.sh 1

I0827 15:55:14.586019 4575 caffe.cpp:351] Avg. Time Taken: 56.9657

- $ sh run-alexnet.sh 2

I0827 15:56:43.421020 4648 caffe.cpp:351] Avg. Time Taken: 41.4431

- $ sh run-alexnet.sh 4

I0827 15:57:29.531169 4721 caffe.cpp:351] Avg. Time Taken: 32.4485

OSU-Caffe Demo: Execution Steps

1. Allocate Nodes and change the host file (use cdcaffe)

2. Go to the root directory

cd /home/awan/hoti17-demo/osu-caffe

3. Setup the library paths

source ./set-caffe.sh

4. Test out the Caffe binary

caffe

5. Run AlexNet training on 2 GPUs (nodes)

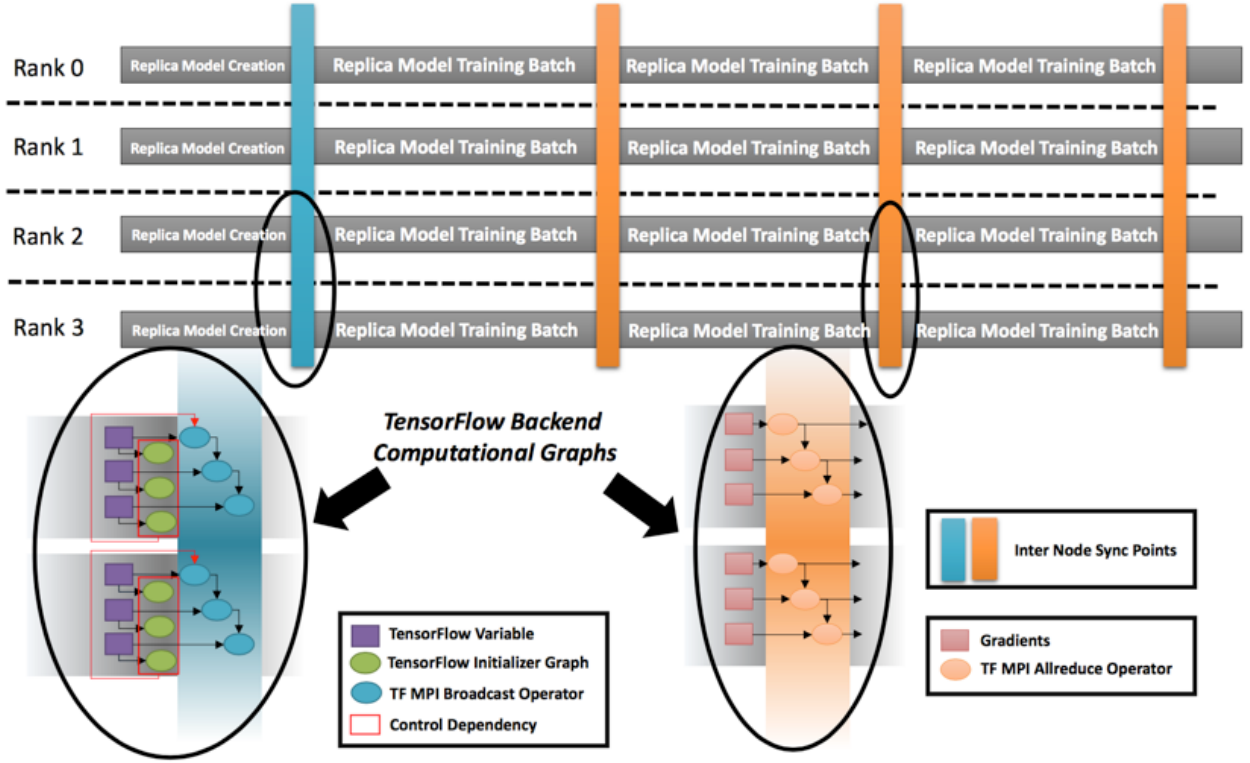
sh run-alexnet.sh 2

Case Studies and Demos: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce (+Demo)
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
  - MPI+NCCL for CUDA-Aware CNTK
  - OSU-Caffe (+Demo)
- **MATEX TensorFlow**
- Scaling DNN Training on Multi-/Many-core CPUs (+Demo)
- PowerAI DDL

MATEX TensorFlow

- MaTEx-TensorFlow is a user-transparent distributed memory implementation of TensorFlow using MPI
  - https://arxiv.org/pdf/1704.04560.pdf
- Automatic parallelization using MPI for GPU and CPU Clusters
- Support for data parallelism – sufficient for majority of DL implementations

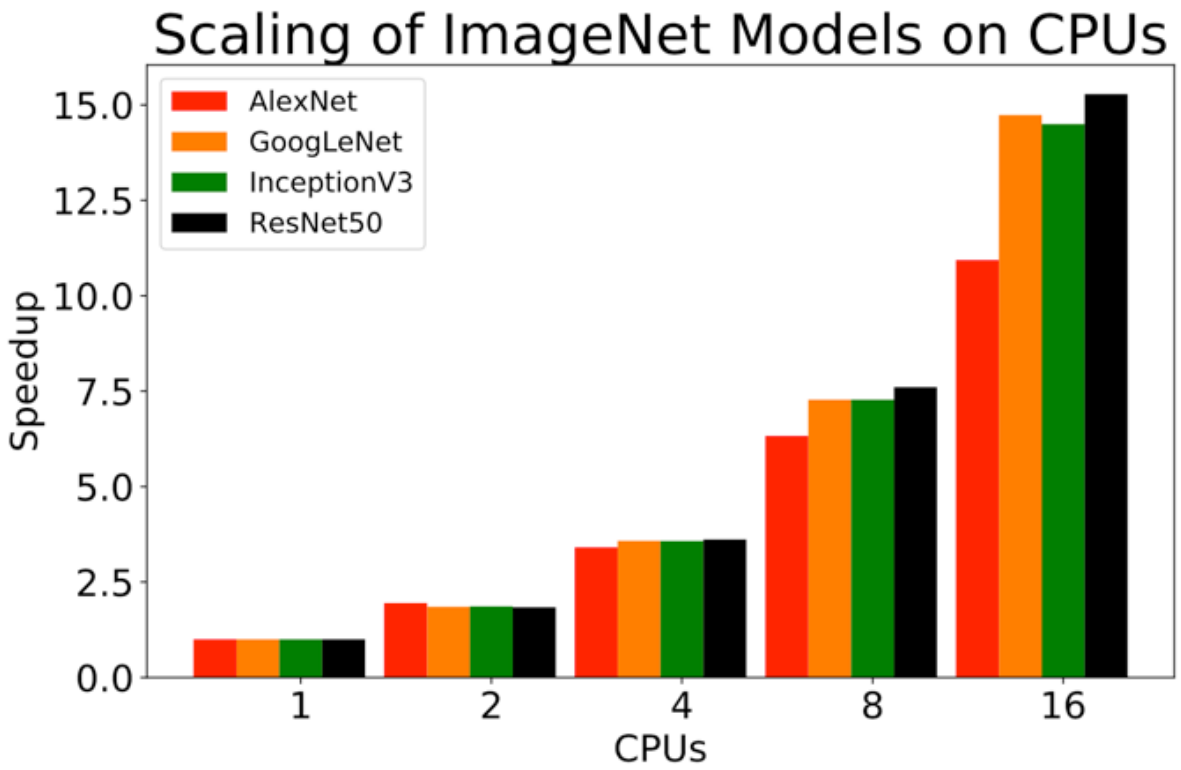


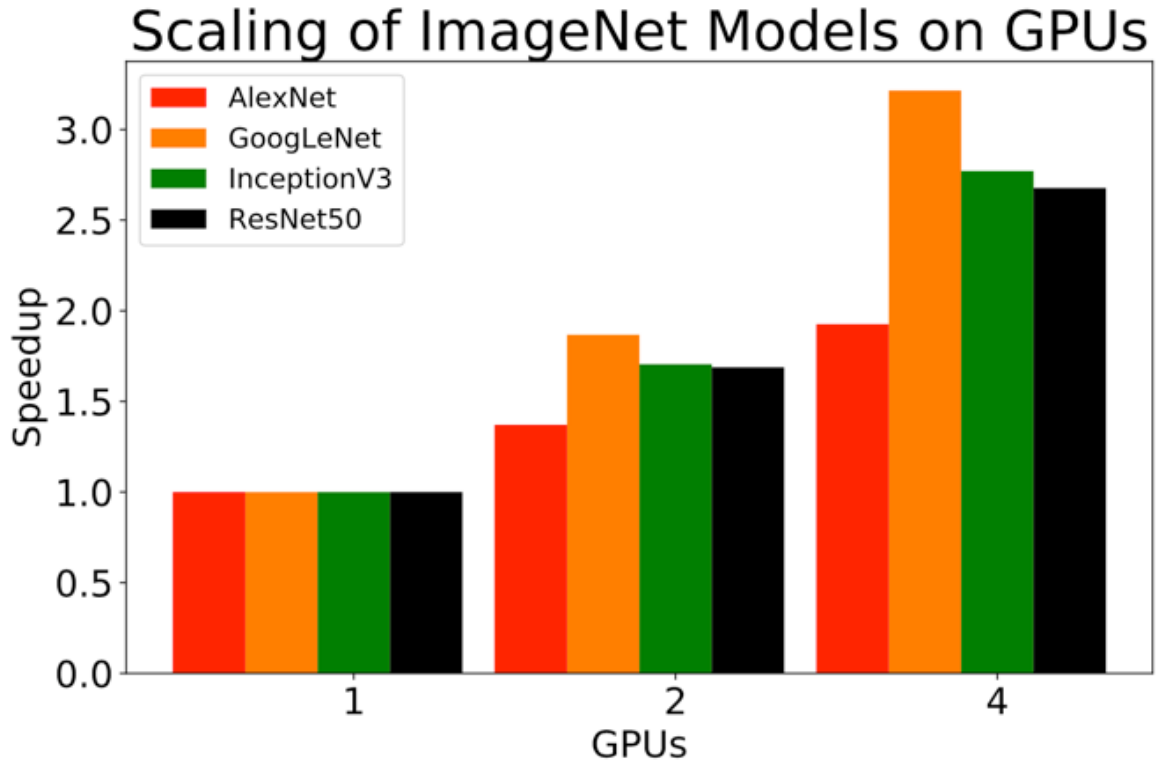
User-transparent Distributed Design

Courtesy: https://github.com/matex-org/matex

MATEX TensorFlow: Scaling and Performance

- Intel Ivy-bridge (20 cores) System is used for CPU-based evaluation
- NVIDIA K-40 GPU has been used



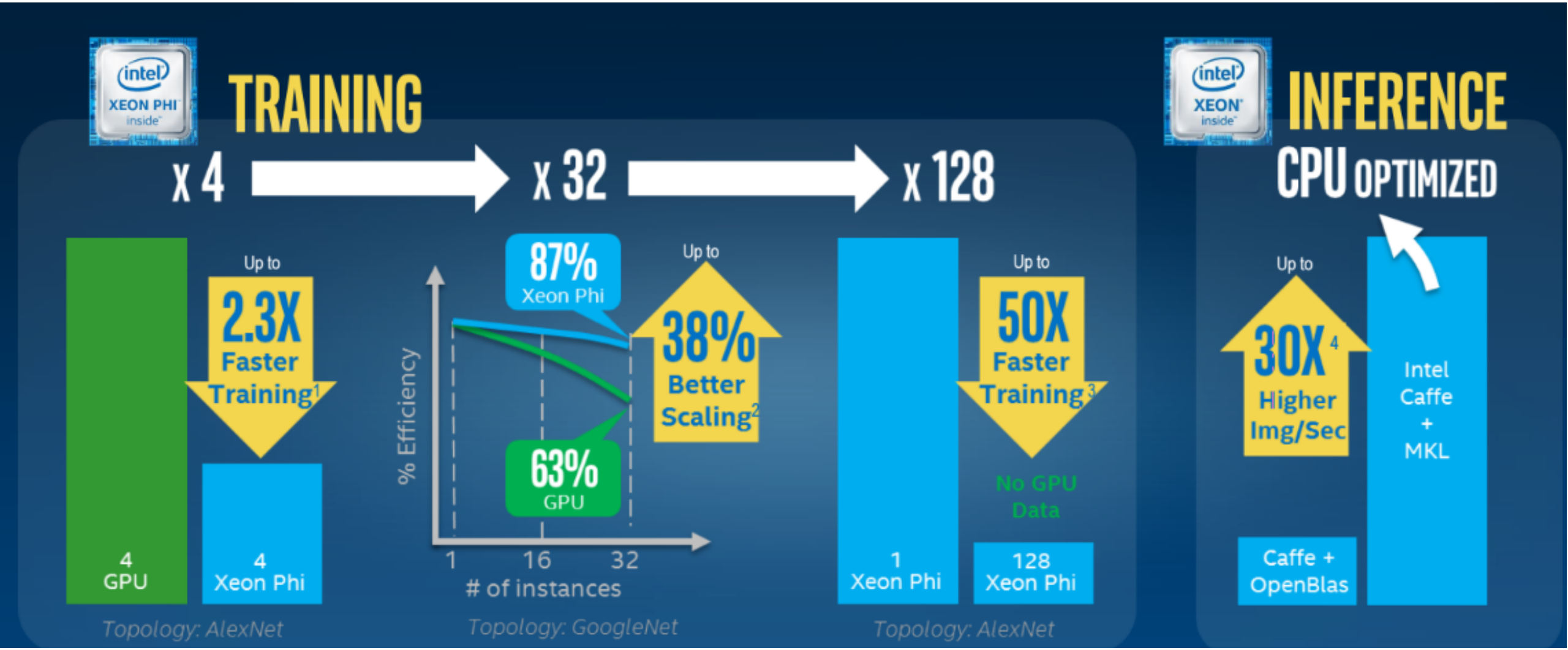


Courtesy: https://arxiv.org/pdf/1704.04560.pdf

Case Studies and Demos: Exploiting HPC for DL

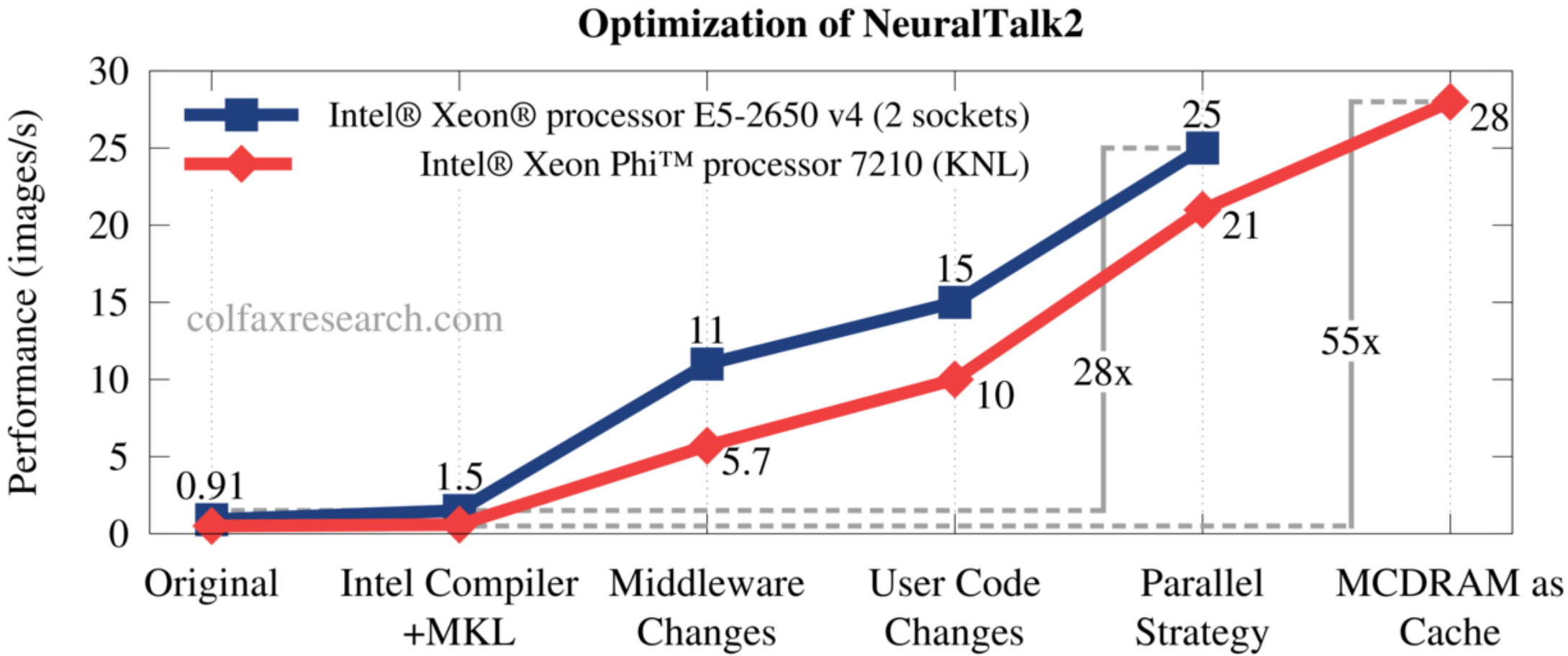
- NVIDIA NCCL
- Baidu-allreduce (+Demo)
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
  - MPI+NCCL for CUDA-Aware CNTK
  - OSU-Caffe (+Demo)
- MATEX TensorFlow
- **Scaling DNN Training on Multi-/Many-core CPUs (+Demo)**
- PowerAI DDL

Optimizing and Scaling DL on Intel CPUs



Courtesy: https://www.nextplatform.com/2016/06/21/knights-landing-solid-ground-intels-stake-deep-learning/

Optimizing NeuralTalk on Intel CPUs with Intel MKL



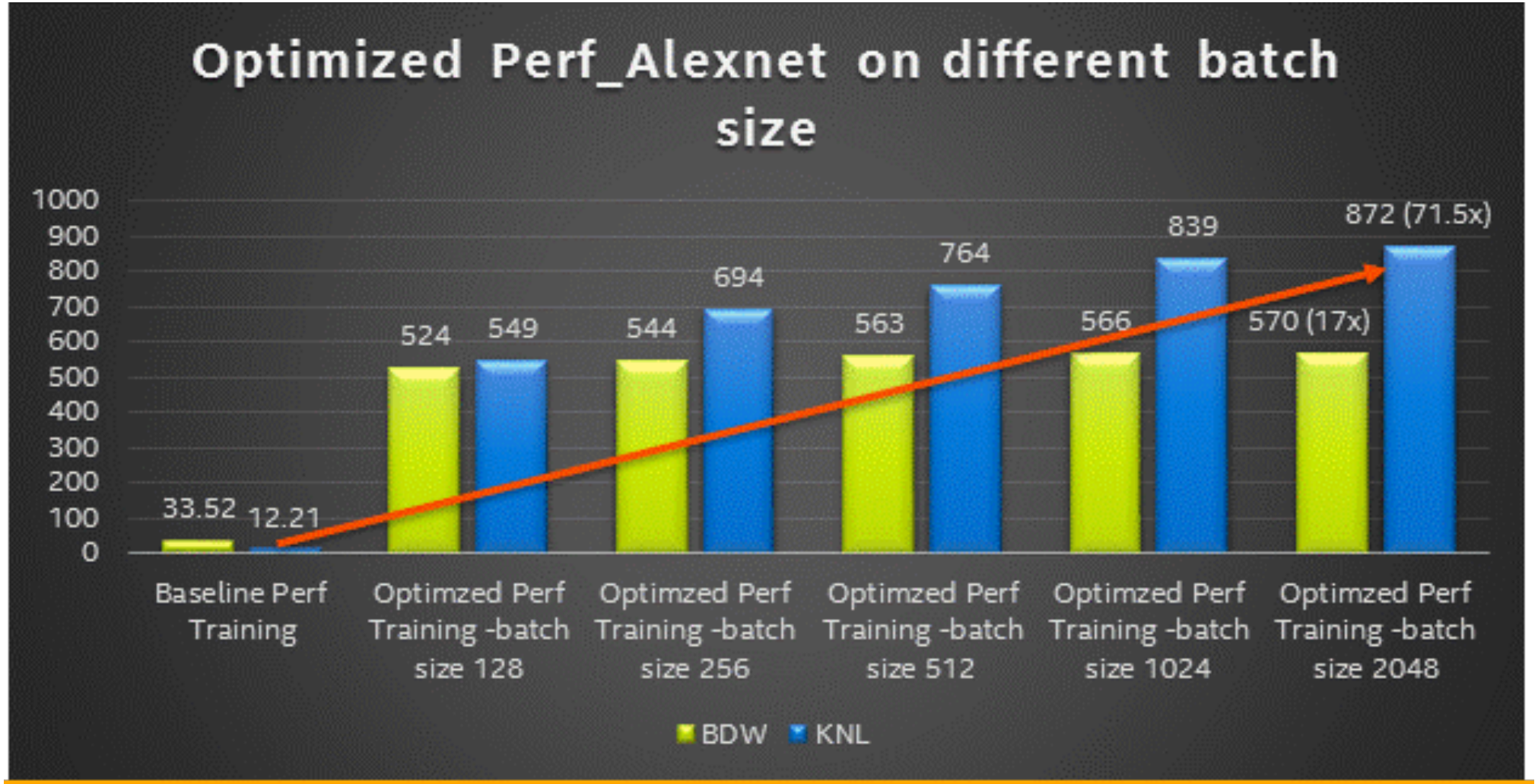
Courtesy: https://colfaxresearch.com/isc16-neuraltalk/

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

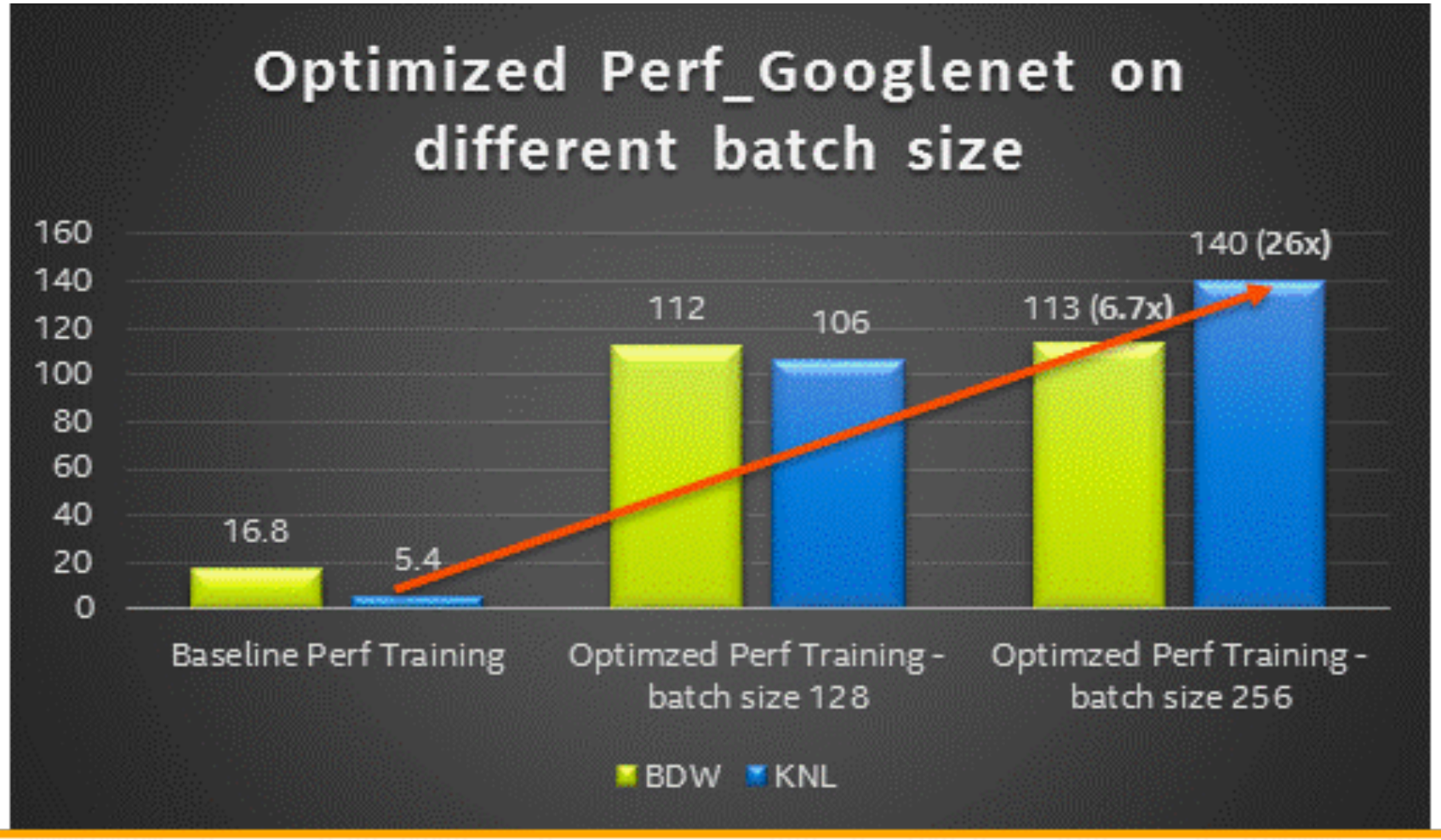
TensorFlow Optimization for Intel CPUs



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

TensorFlow Optimization for Intel CPUs



26x Speedup From New Optimizations – available through Google's TensorFlow Git

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

Demo: Running Intel-Caffe with Intel-MLSL

- There are certain versions of Intel-MLSL (intel/mlsl_2017.0.014/) and a specific git commit (or release tag) that will work
  - It may vary according to your system as well so some hit-and-trial will be required
  - Weak scaling is the default mode for Intel-Caffe

```
commit b0ef3236528a2c7d2988f249d347d5fdae831236
Author: Haihao Shen <haihao.shen@intel.com>
Date:   Wed May 17 11:19:56 2017 +0800

    Upgrade MKL-DNN to version ecf1883a94239a19d442356d32e1076a15b88e7a
```

- **Steps to execute (2 process runs sometimes hang)**

cd /home/awan/projects/intel-apr17-caffe-mlsl

sh run-mpi.sh 4

Demo: Performance Improvement using Intel MKL2017

- Intel-Caffe provides extra options with the Caffe binary
  - run with and without –engine CAFFE or –engine MKL2017/MKLDNN
  - -engine MKL2017 and -engine MKLDNN provide similar performance
  - Originally, the CPU code in Caffe was heavily under-optimized
    - Single threaded
    - No DNN-specific optimizations
    - -engine CAFFE is multi-threaded (OpenMP) Berkeley's Caffe design
    - -engine MKL2017/MKLDNN have both multi—threading and optimized convolutions
- Example run commands:
  - caffe time -model models/bvlc_alexnet/train_val.prototxt -engine CAFFE
  - caffe time -model models/bvlc_alexnet/train_val.prototxt -engine MKL2017

Intel-Caffe Demo: Caffe time() Benchmark Comparison

- -engine CAFFE

```
I0828 02:19:51.786986 22535 caffe.cpp:607] Average Forward-Backward: 1171.24 ms.
I0828 02:19:51.786993 22535 caffe.cpp:610] Total Time: 58562 ms.
I0828 02:19:51.787001 22535 caffe.cpp:611] *** Benchmark ends ***
```

- -engine MKL2017

```
I0828 02:14:43.755236 22444 caffe.cpp:607] Average Forward-Backward: 686.1 ms.
I0828 02:14:43.755242 22444 caffe.cpp:610] Total Time: 34305 ms.
I0828 02:14:43.755251 22444 caffe.cpp:611] *** Benchmark ends ***
```

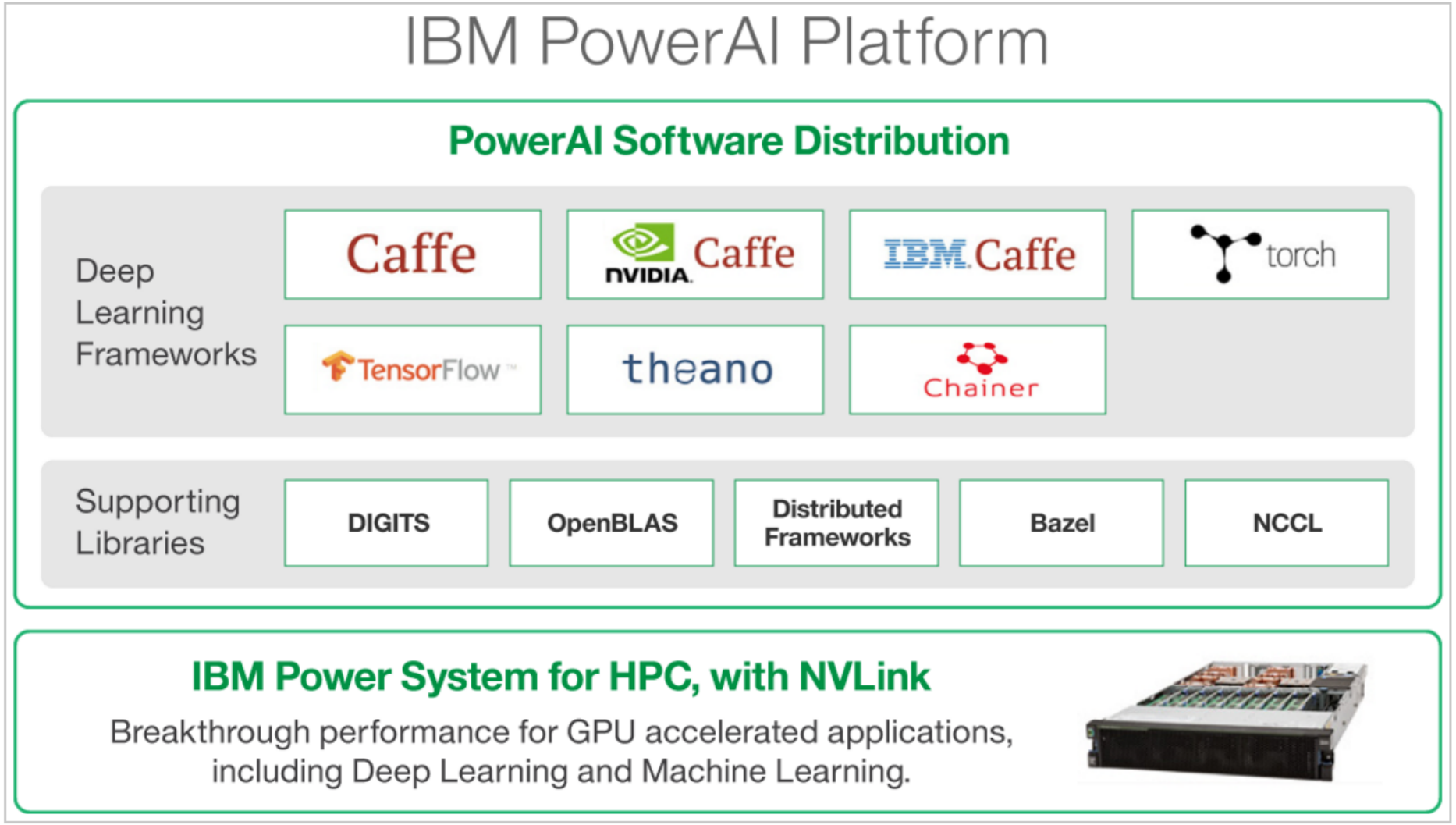
- -engine MKLDNN

```
I0828 02:18:20.974902 22498 caffe.cpp:607] Average Forward-Backward: 741.2 ms.
I0828 02:18:20.974911 22498 caffe.cpp:610] Total Time: 37060 ms.
I0828 02:18:20.974917 22498 caffe.cpp:611] *** Benchmark ends ***
```

Case Studies and Demos: Exploiting HPC for DL

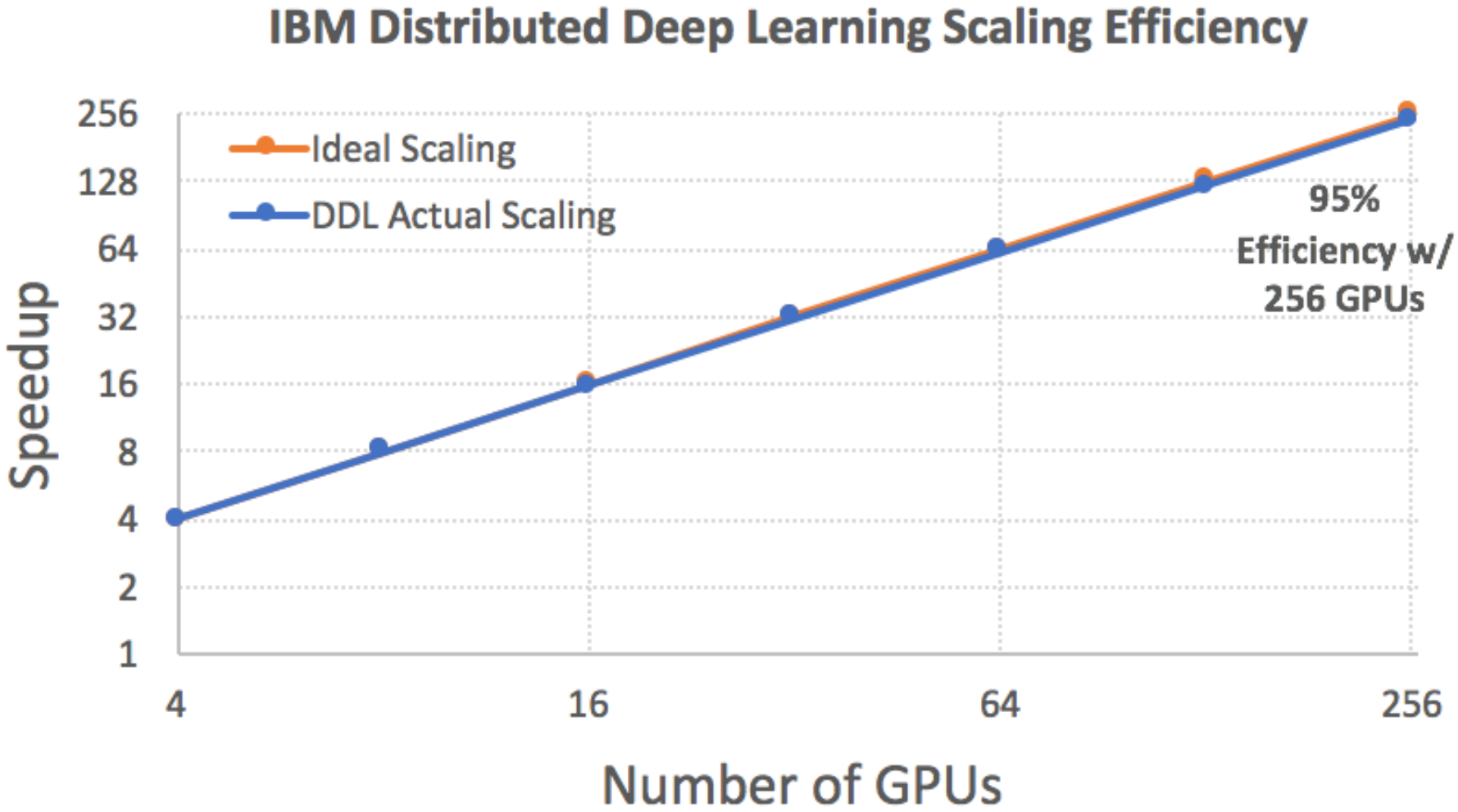
- NVIDIA NCCL
- Baidu-allreduce (+Demo)
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
  - MPI+NCCL for CUDA-Aware CNTK
  - OSU-Caffe (+Demo)
- MATEX TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs (+Demo)
- **PowerAI DDL**

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
- Case Studies and Demos
- **Open Issues and Challenges**
- Conclusion

Open Issues and Challenges

- Which Framework should I use?
- Convergence of DL and HPC
- What is the rationale behind NCCL and Gloo?
- DL Benchmarks and Thoughts on Standardization
- Scalability and Large batch-size training?

Which Framework should I use?

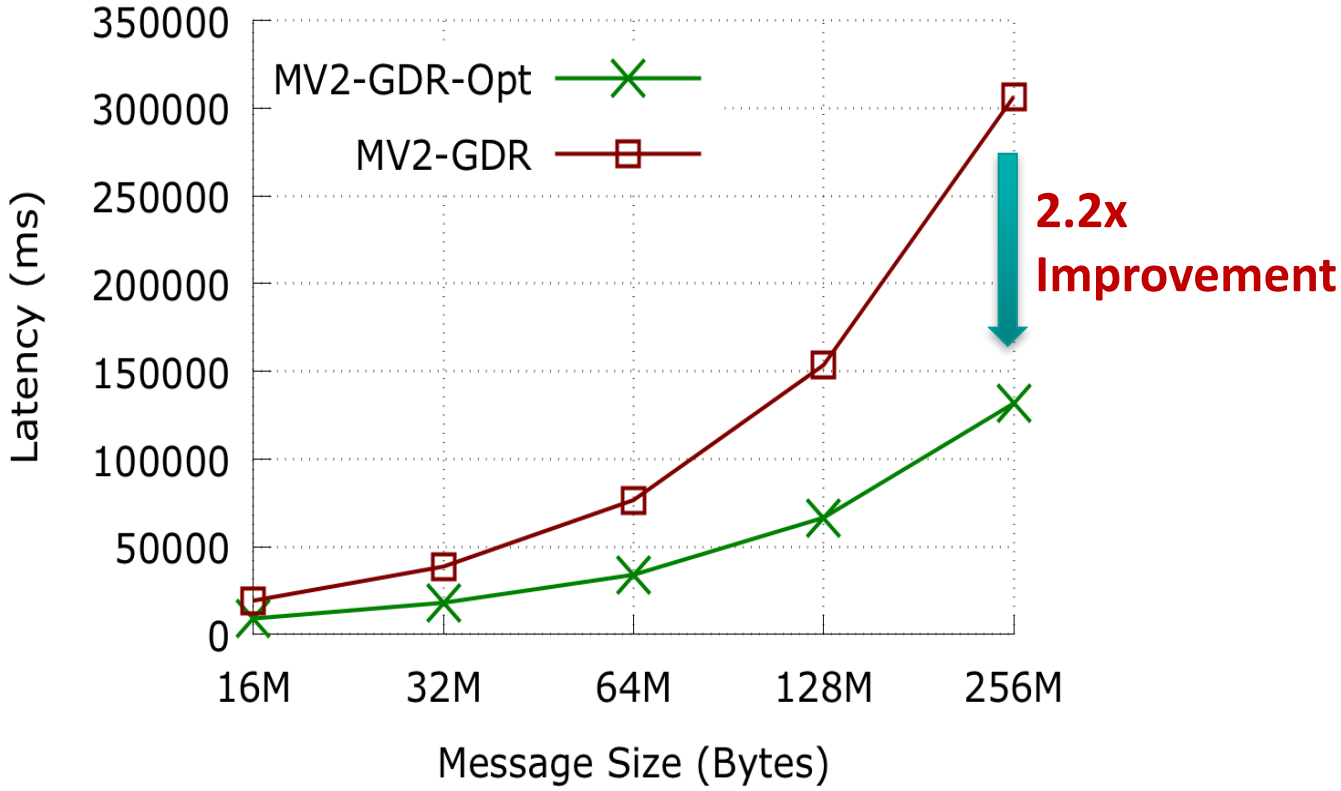
- Depends on the higher-level Application needs
  - Image, Speech, Sequences, etc.
- Depends on the hardware availability
  - GPUs are good in general
  - If you have Intel CPUs, Intel-Caffe and Intel-optimized TensorFlow are a good start
- Also depends upon your programming knowledge and requirements
  - Python frontend or C++ frontend?
  - Model designer tools needed?
    - Keras can use other DL frameworks as a back-end and provides a high-level interface.

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

What is the rationale behind NCCL and Gloo?

- General-purpose MPI vs Special-purpose NCCL/Gloo?
  - Both approaches have certain trade-offs
- MPI works great for most cases but not all
  - Can we do something about it? [1], [2]
- Can NCCL or Gloo be used without MPI?
  - Intra-node --> Clearly, Yes!
  - Inter-node --> Not very clear!
    - Some Rendezvous mechanism is needed
    - Job-launch will have issues without MPI
      - especially for very large scale runs
    - What can MPI not do that NCCL and Gloo can?
    - What about point-to-point communication?



1. A. A. Awan, K. Hamidouche, A. Venkatesh, and D. K. Panda, Efficient Large Message Broadcast using NCCL and CUDA-Aware MPI for Deep Learning. In *Proceedings of the 23rd European MPI Users' Group Meeting* (EuroMPI 2016). [Best Paper Nominee]

2. 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)

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

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

Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
- Case Studies and Demos
- 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
    - MATEX-TensorFlow
    - Intel-Caffe and Intel-MLSL
- Many other open issues
- Need collaborative efforts to achieve the full potential performance
- Standardization may help remove fragmentation in DL frameworks

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

- 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.)
- N. Islam (Ph.D.)
- M. Li (Ph.D.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

Current Research Scientists

- X. Lu
- H. Subramoni

Current Post-doc

- A. Ruhela

Current Research Specialist

- J. Smith
- M. Arnold

Past Students

- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- 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.)
- 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.)

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

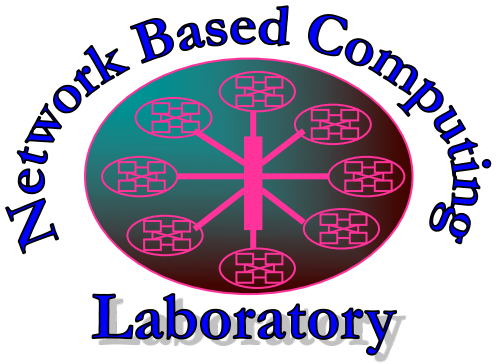
- D. Bureddy
- J. Perkins

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/