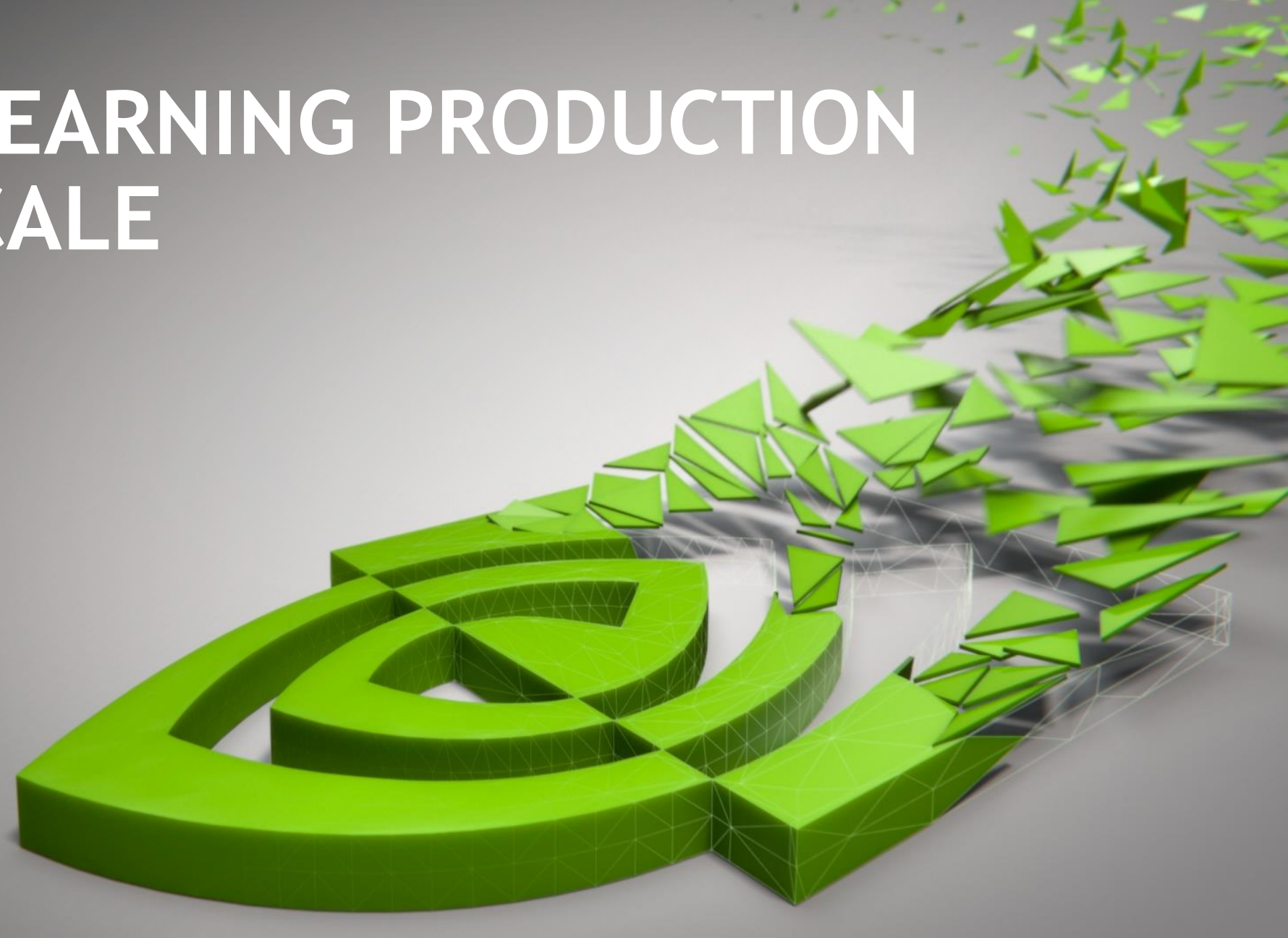
DEEP LEARNING PRODUCTION AND SCALE

Mike Houston

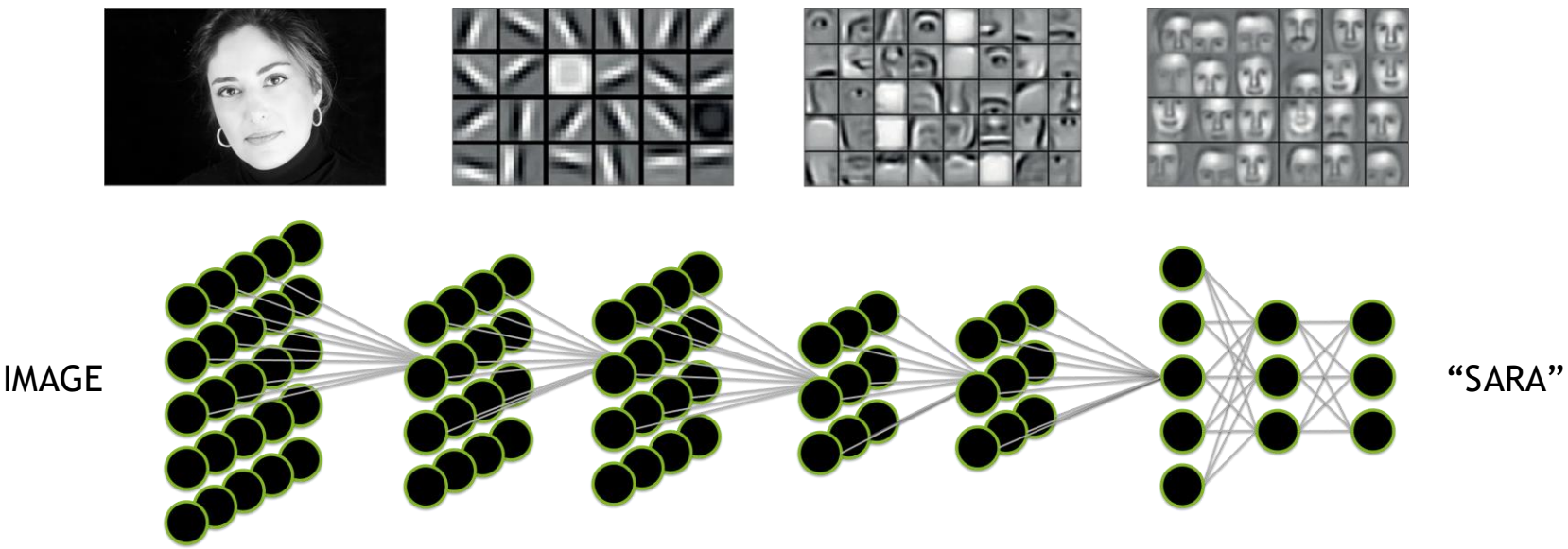
NVIDIA



Deep Learning has changed the way we think about developing software

WHERE IT ALL STARTED

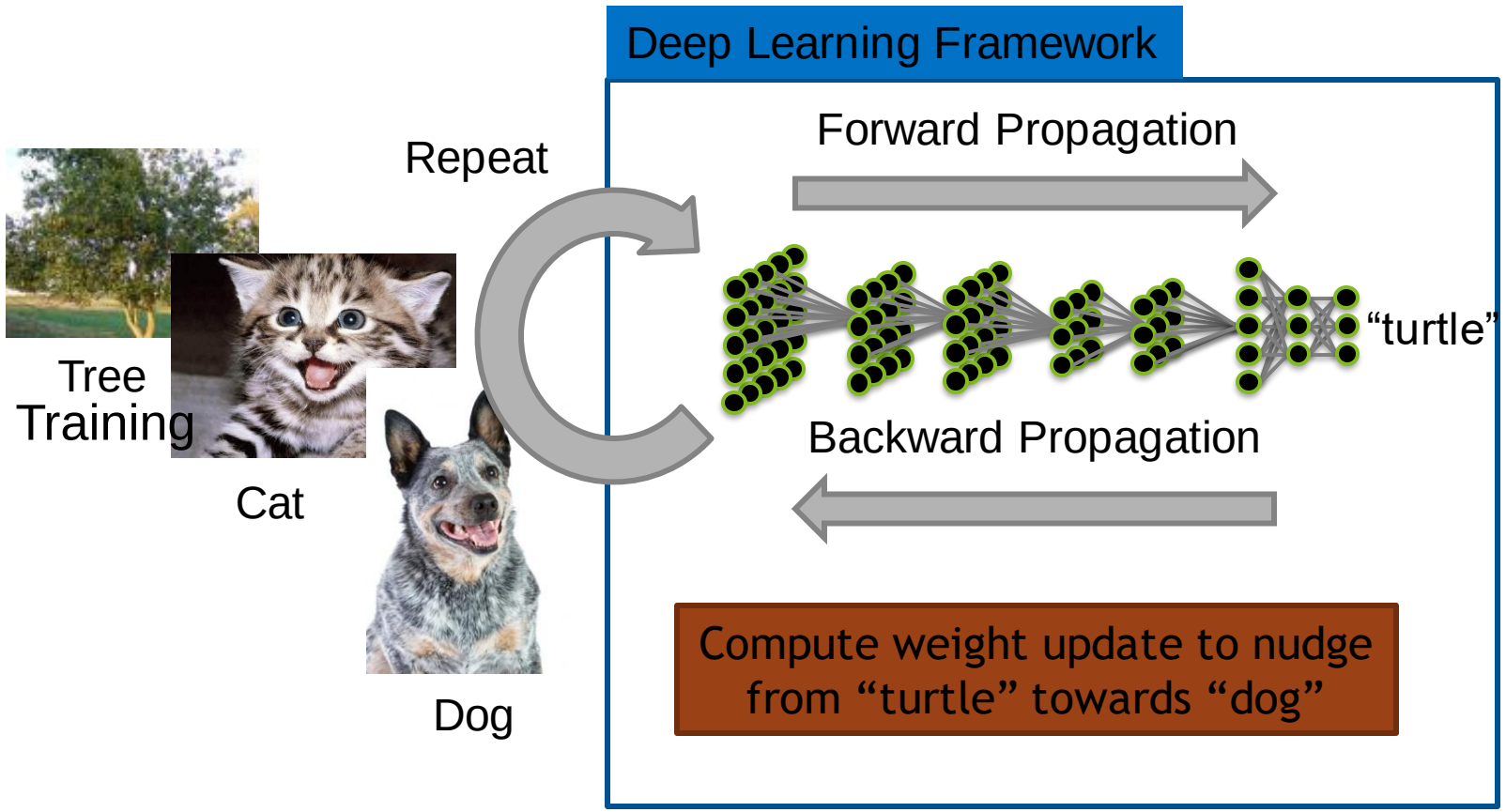
The magic algorithm from the 80s



"Neural nets are universal approximators, convolutions a good prior for spatial data… there seems to be no limit to what these CNNs can learn!"

— Someone in Yann LeCun's lab, circa 2010

DL TRAINING 101



2011, 2012, 2013…

Dreams became reality

Compute power finally enough to train DNNs at scale

Simplified DNN architectures (ReLU) and smart regularization tricks (Dropout)

Research results exploded, beating several tough perception challenge benchmarks

Timescales started to change…

2017

Deep Learning is now (almost) a commodity

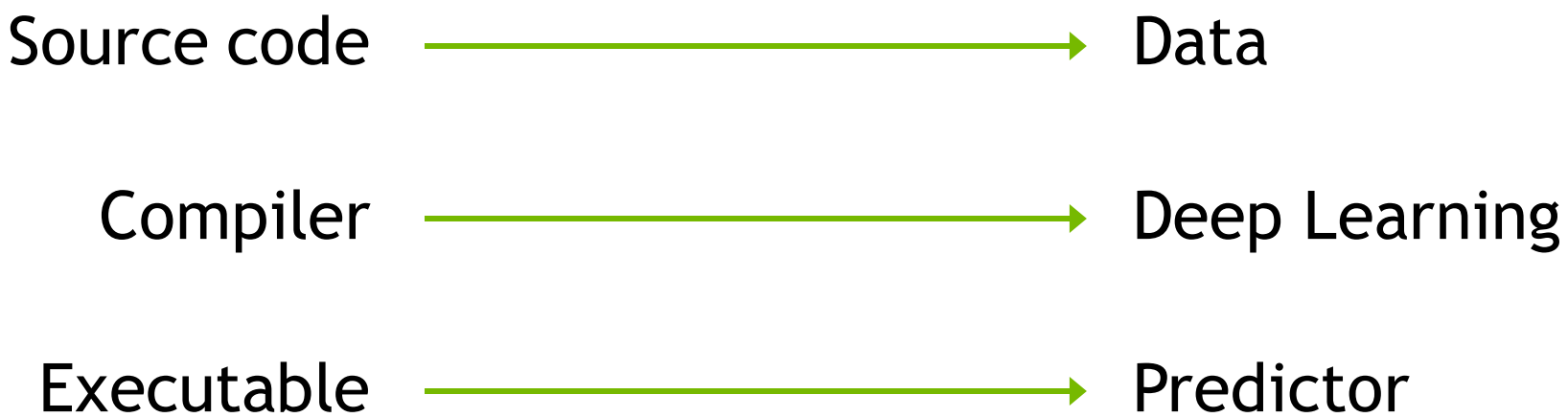
Used as a building block to improve many baseline systems, for ads, content recommendation, image/video search, speech recognition, autonomous vehicles…

Results quickly transitioning from research to production

Many self-taught engineers are now using Deep Learning frameworks and methods to add value to their products

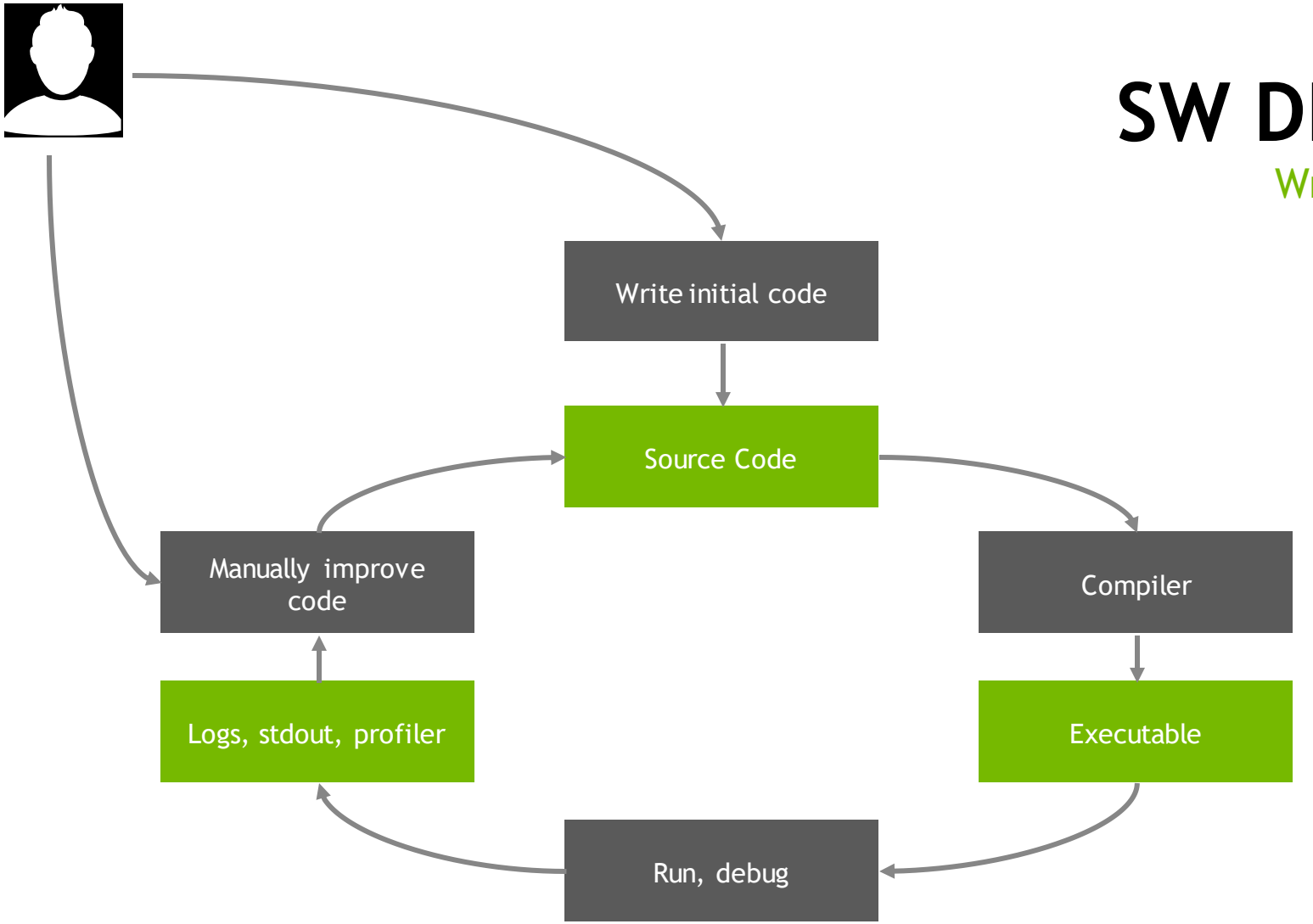
Deep Learning has changed the way we think about developing software…

… we need to *actually* change the way we develop software!


Source code
Data
Compiler
Deep Learning
Executable
Predictor

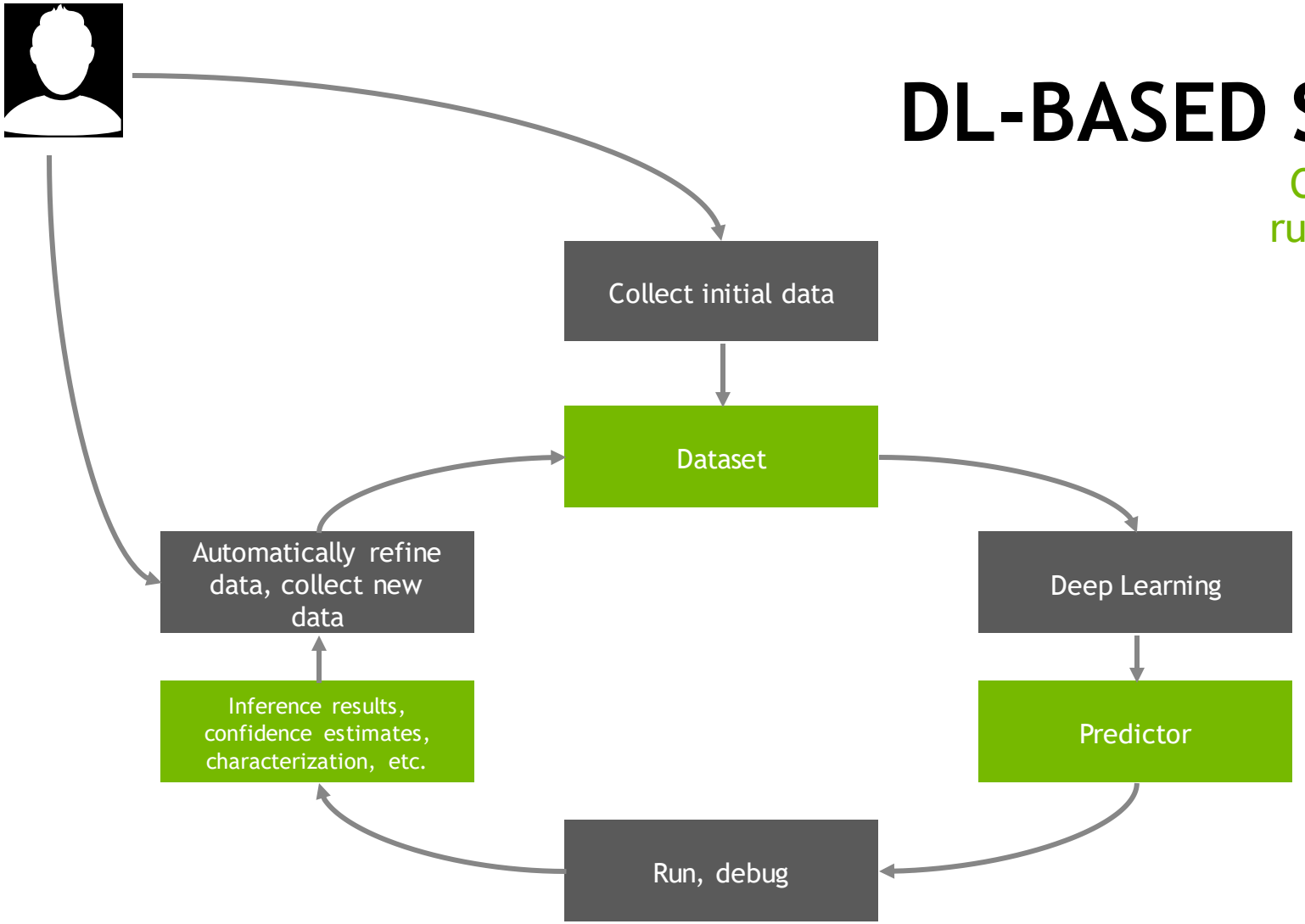
SW DEV PROCESS

Write code, compile, test, debug, repeat…



DL-BASED SW PROCESS

Collect initial data, train, run/debug, mine new data



DL-BASED SOFTWARE PROCESS

Methods & tools required

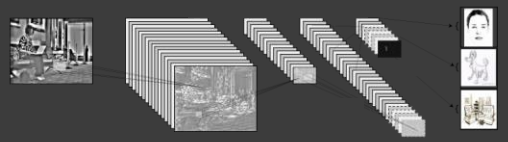
1. Given a fixed dataset, find the best predictor
    a) Explore best architectures/models (meta-optimize) — large-scale map jobs
    b) Fit one model give meta-params — multi-GPU/node, high-bandwidth job
2. Given a fixed predictor, find the next best data
    a) Mine/analyze raw, unlabeled data — large-scale inference jobs
    b) Store datasets, models, experiment data — asset/version management system
3. Deploy
    a) Convert end-to-end inference pipeline — exporter/converter/compiler to target env

DL FOR CARS

PBs of data, large-scale labeling, large-scale training, etc.

POST /datasets/{id}

Inference optimized DNN (TensorRT)



Datasets

Labels

Manually selected data

Labeling

Deep Learning

Train/test data

Metrics

Simulation, verification results

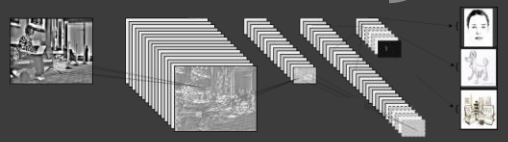
DL FOR CARS

Active learning strategies to meet business needs

POST /datasets/{id}

Trained Models

Inference optimized DNN (TensorRT)



Deep Learning

Train/test data

Datasets

Mine highly confused / most informative data

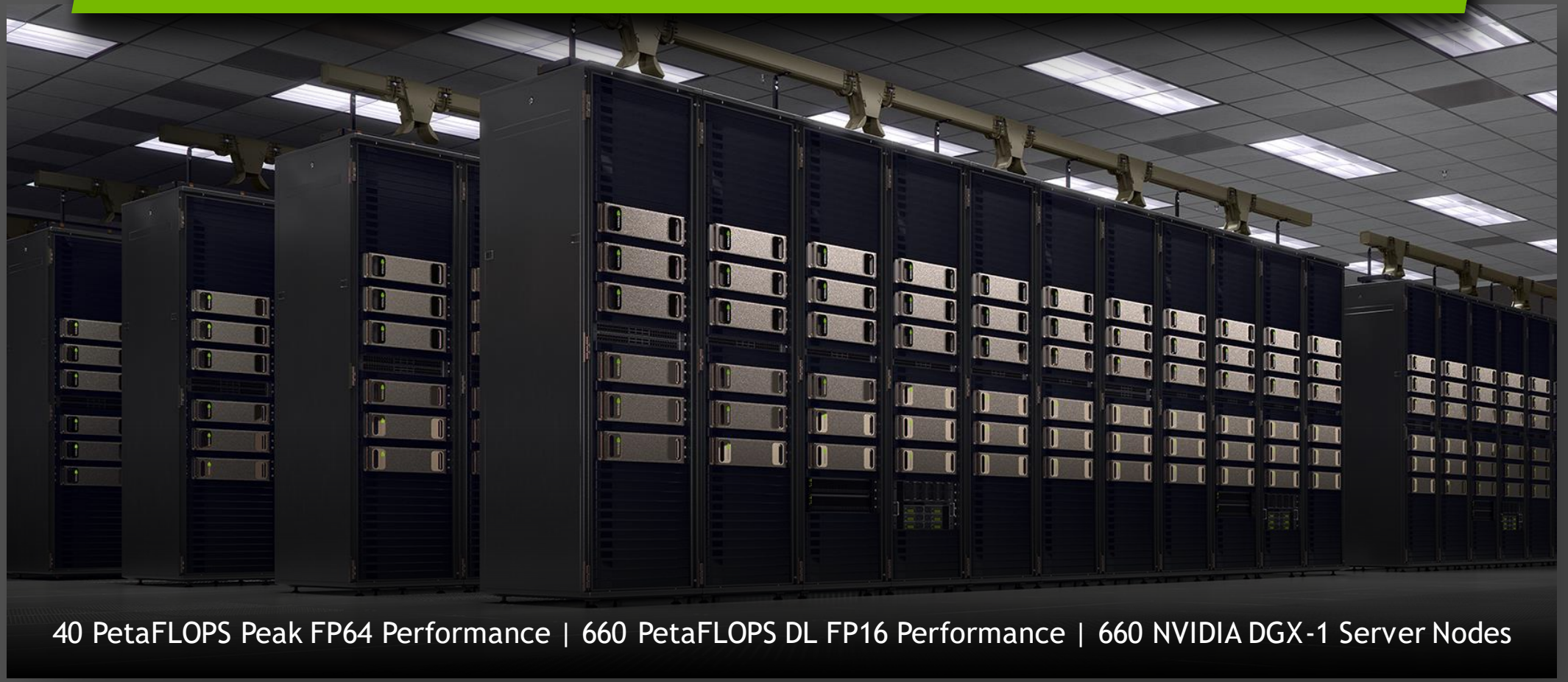
Intelligently selected data

Labels

Labeling

BUILDING AN AI SUPERCOMPUTER

ANNOUNCING
NVIDIA SATURNV WITH VOLTA



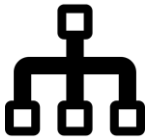
40 PetaFLOPS Peak FP64 Performance | 660 PetaFLOPS DL FP16 Performance | 660 NVIDIA DGX-1 Server Nodes

KEY FINDINGS

Learnings from building the world's most energy-efficient AI supercomputer

DNN's are inherently more stressful than HPC and other workloads

Plan for higher density power per rack, and associated cooling demand

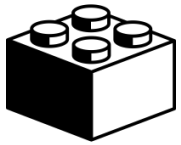


Implement a non-blocking networking fabric

InfiniBand networking deployed in a fat-tree topology, with a minimum of 2 connected InfiniBand EDR ports per node

Storage must support very large datasets with read-heavy I/O

Local SSD/NVMe cache for accelerated reads. NFS appliances with 10GbE for small cluster configs. Parallel FS starting at medium clusters, 32+ nodes. Deep tiering and caching for large scale



Use proven config. models for easy DNN scalability

Modular designs, easily replicable, predictable performance. 36-node pods for large systems

SUPER-REAL-TIME SIMULATION

Accelerating Development, Making Autonomous Vehicles Safer

Self-driving technologies continually re-evaluated to surpass safety of human-driven

Simulation evaluates dangerous, uncommon scenarios

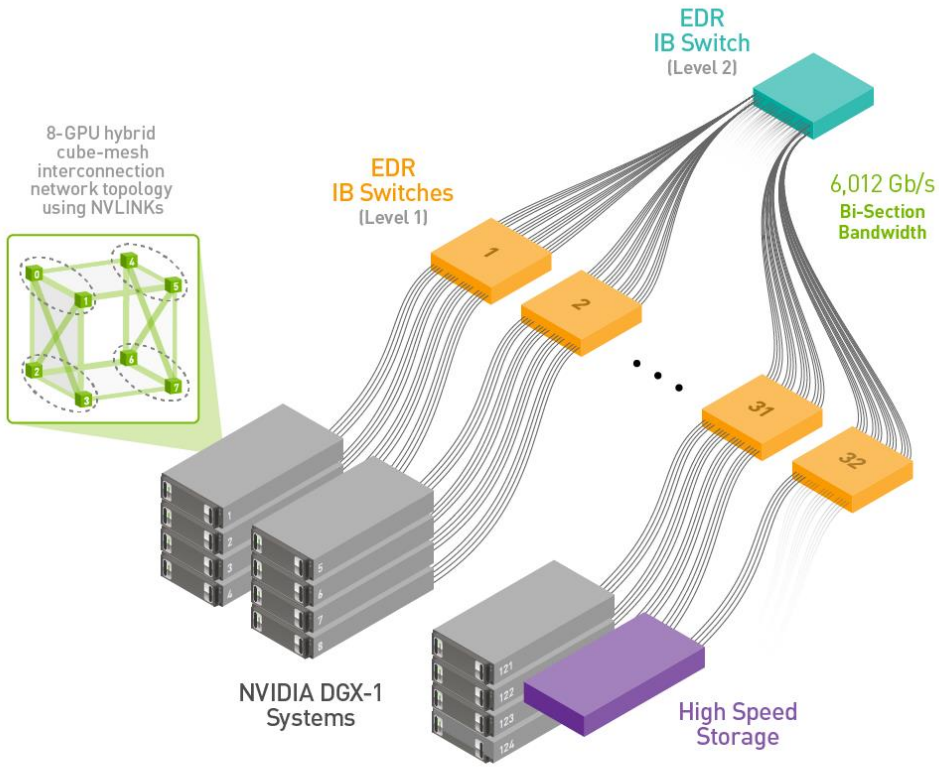
Super real-time sim = 300,000 miles driven in 5 hours on 8 SaturnV nodes

Every paved road in the U.S. in just 2 days

NVIDIA

NETWORKING TOPOLOGY

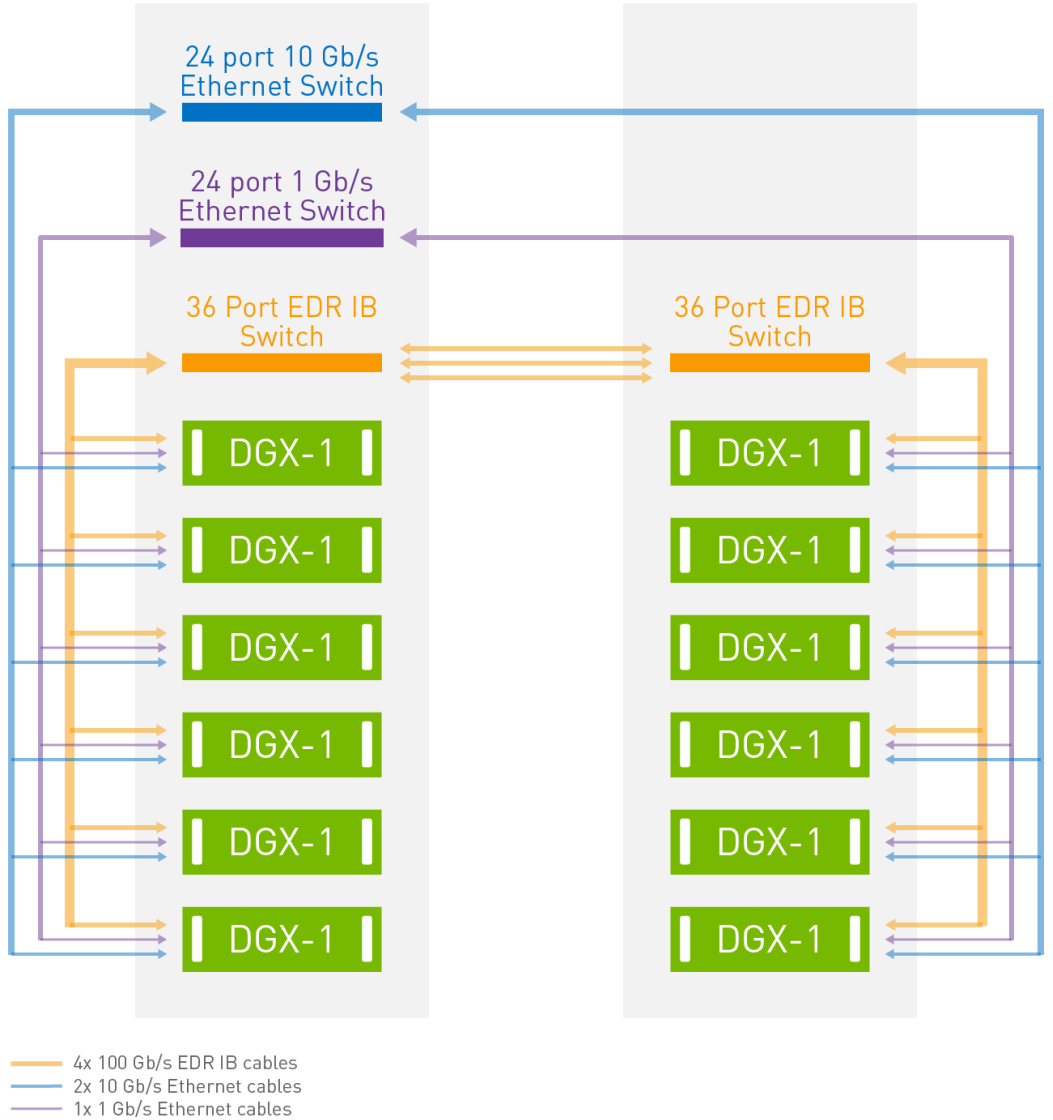
- Ingest data as fast as possible
- Pass data rapidly between nodes across cluster
- Similar to HPC networking architecture
- InfiniBand = ultra high bandwidth, low latency, collision free
- Two-tier network with root and leaf switches
- Any to Any connectivity with full bi-section bandwidth & minimum contention between nodes



SMALL CLUSTER

up to 12 DGX-1 nodes

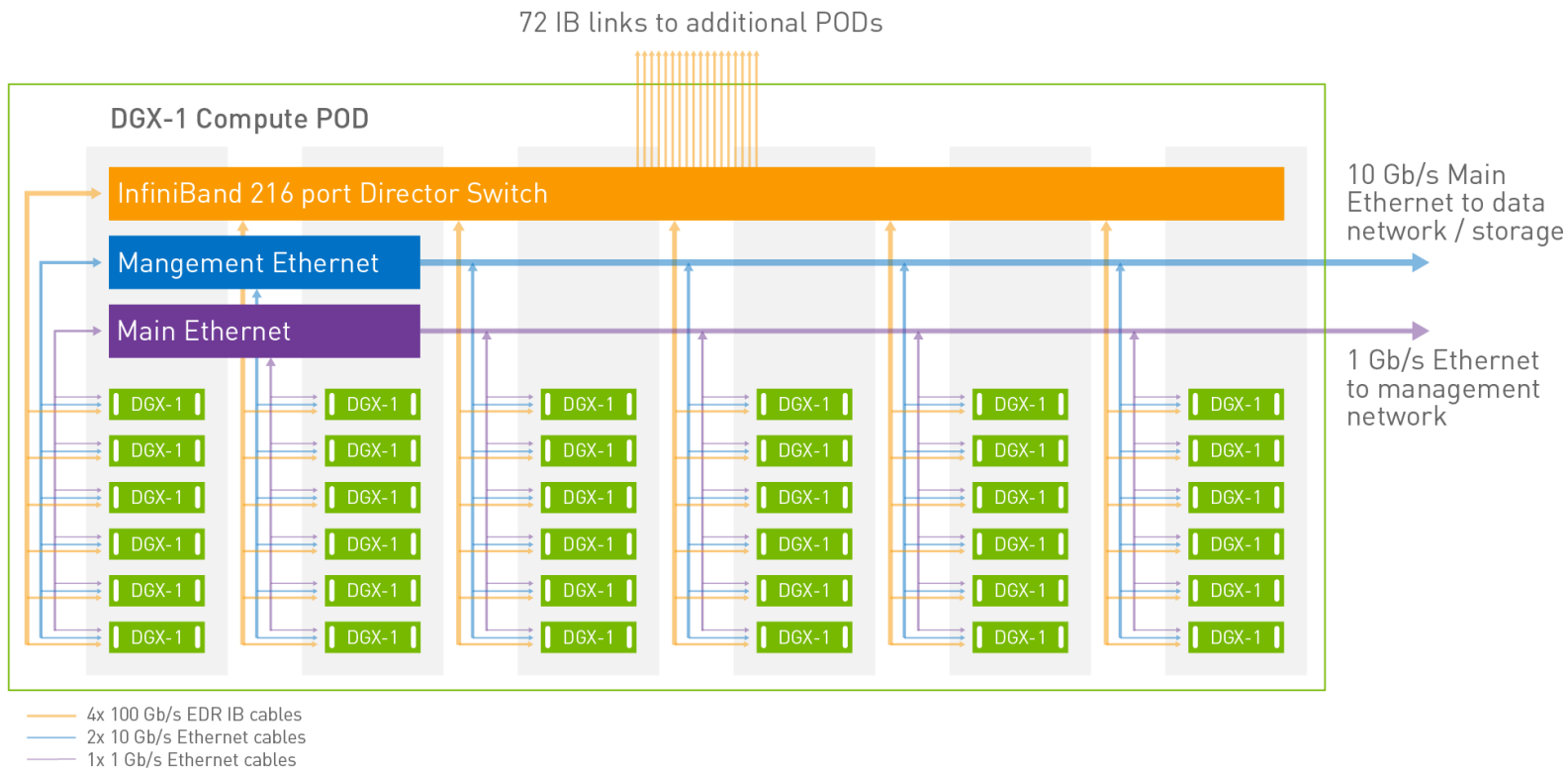
- Assume growth for up to 12 nodes
- 2 racks, 2 IB switches (36 ports)
- 19.2 kW per rack, but can split across racks if necessary
- Full bi-section bandwidth for each group of 6 nodes
- 2:1 oversubscription between groups of 6



MEDIUM CLUSTER

up to 36 DGX-1 nodes

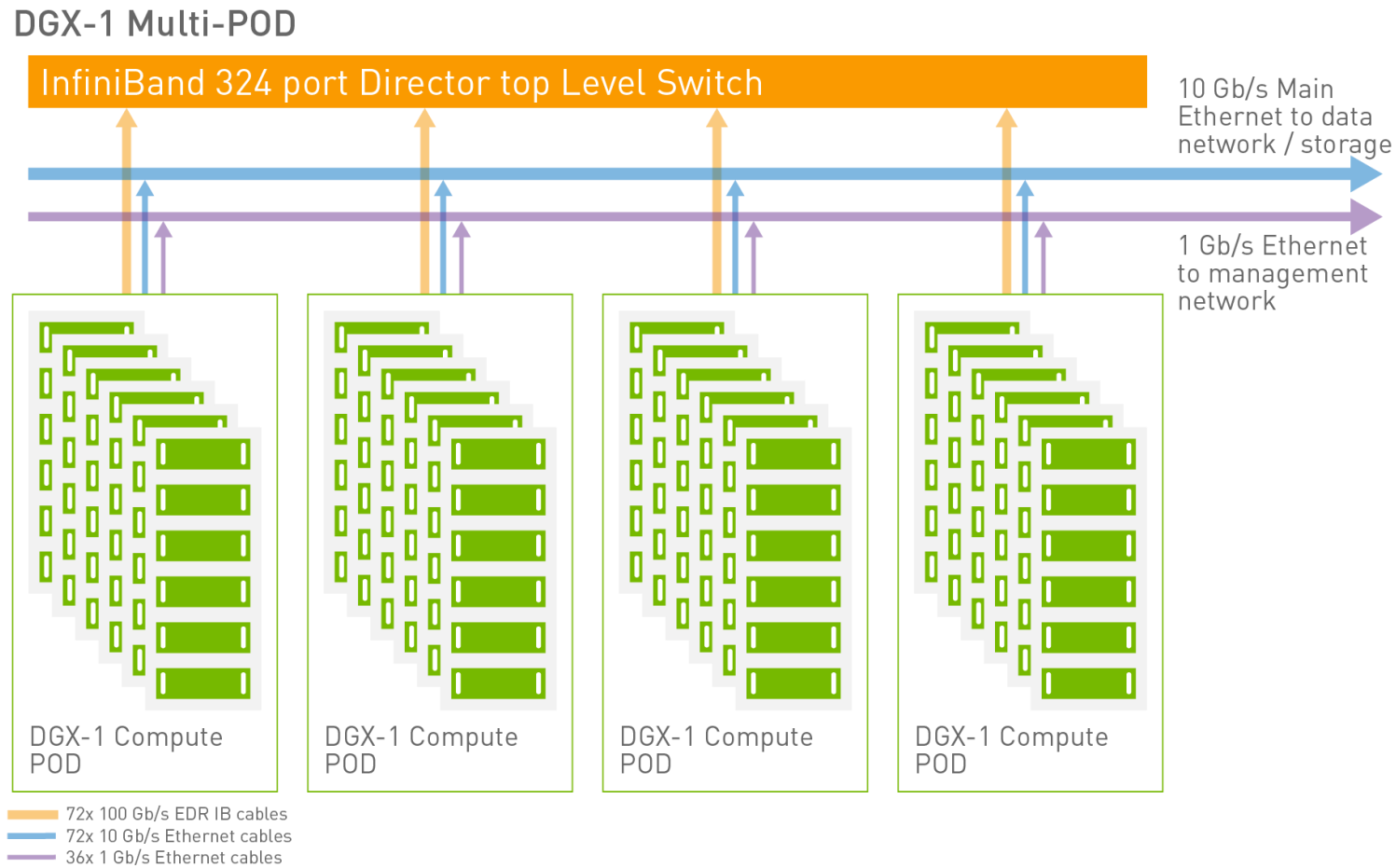
- Defines a DGX-1 “POD”
- Can be replicated for greater scale, ex: large cluster configuration
- 6 racks, 6 nodes per rack
- Larger IB director switch (216 ports) with capacity for more pods via unused ports



LARGE CLUSTER

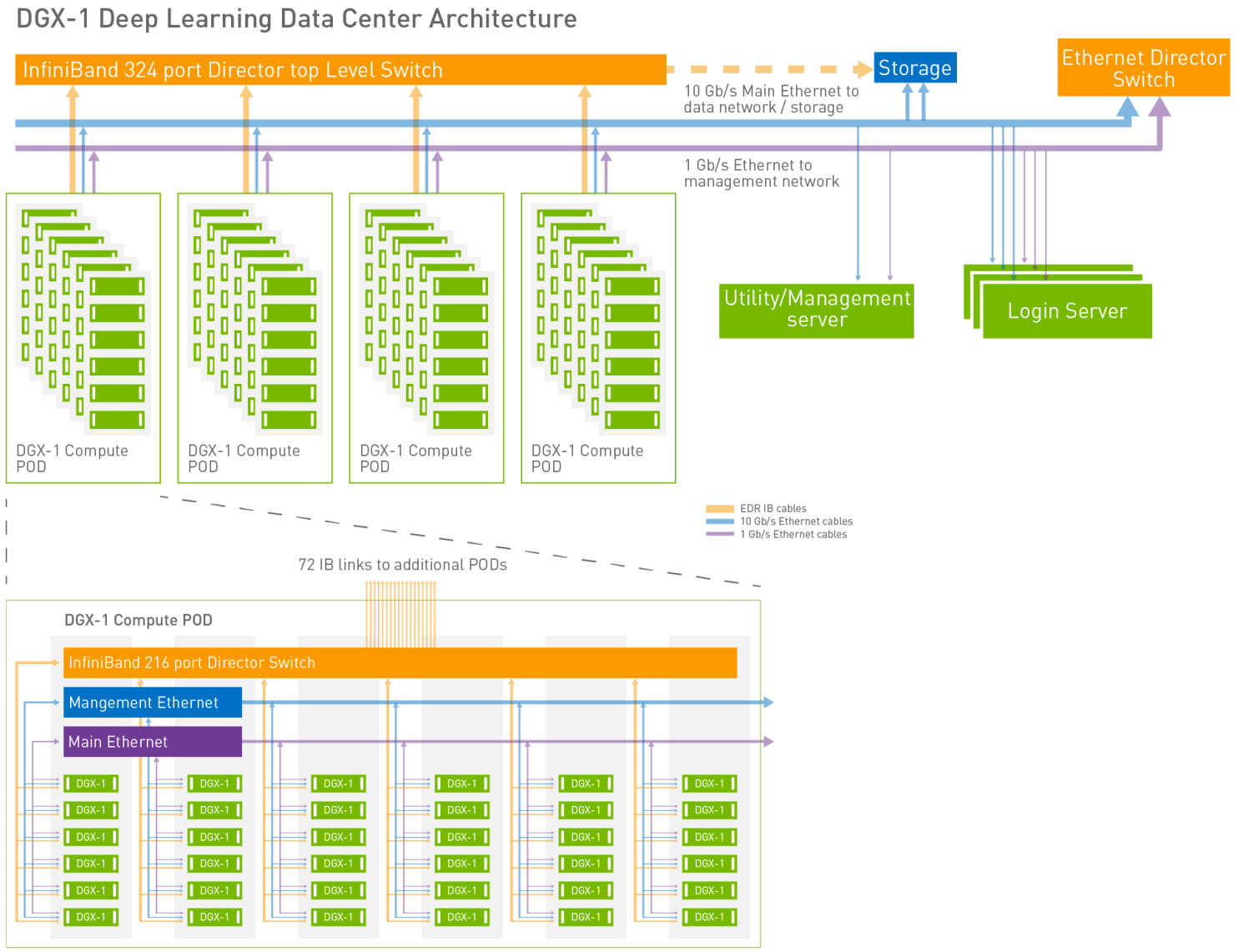
up to 144 DGX-1 nodes (4 ”PODS”)

- Implements 4 DGX-1 pods
- Distributed across 24 racks
- Full bi-section bandwidth within pod, 2:1 between pods
- Training jobs ideally scheduled within a pod, to minimize inter-pod traffic



DGX-1 DEEP LEARNING DATA CENTER

Reference Architecture



“A-HA” MOMENTS IN DL CLUSTER DESIGN

Additional design insights to get you started

Overall Cluster	Rack Design	Networking	Storage	Facilities	Software
• HPC similar to DL • HPC expertise can help in design • Even with HPC, the similarities are limited	• DL drives close to operational limits; • Assume less headroom • Proper airflow is crucial to cluster performance	• Like HPC, InfiniBand is preferred • high bandwidth, low latency • Maximize per-node IB connections	• Datasets range from 10k’s to millions objects • Petabyte levels of storage • Large variance • 90:1 Read:Write	• GPU data center operates at near-max power • Assume higher watts per-rack than most enterprise datacenter • dramatically higher FLOPS/watt = floorspace saved	• Scale requires “cluster-aware” software • NCCL2 = GPU/multi-node acceleration • Automatic topology detect • DL framework optimizations

TRAINING AT (MEDIUM) SCALE

Or, how does one make use of 1024 GPUs?

Data parallel training*

Asynchronous stochastic gradient descent

Model parallel training*

Hybrid parallel training*

Hyperparameter sweep

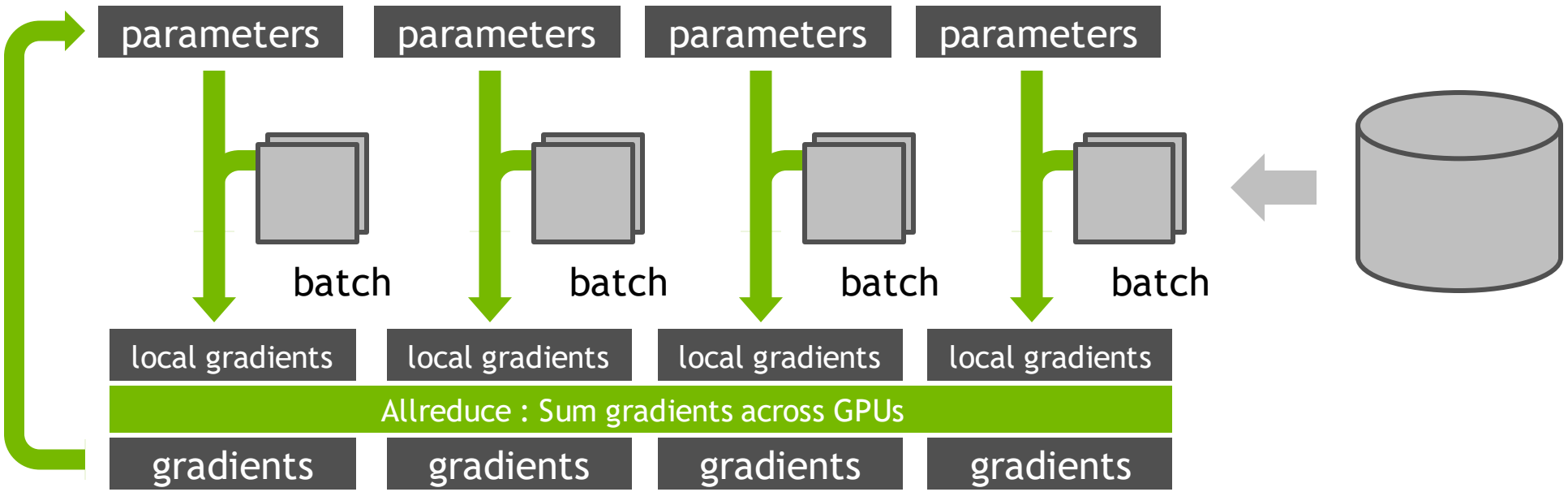
Ensemble/MoE training

***One Weird Trick For Parallelizing Convolutional Neural Networks**
https://arxiv.org/abs/1404.5997

DATA PARALLEL

DNN TRAINING ON MULTIPLE GPUS

Making DL training times shorter



Data parallelism : split batch across multiple GPUs

IMPLEMENTATIONS

AlexNet – “2 Tower” to run on 2GPUs – 2012

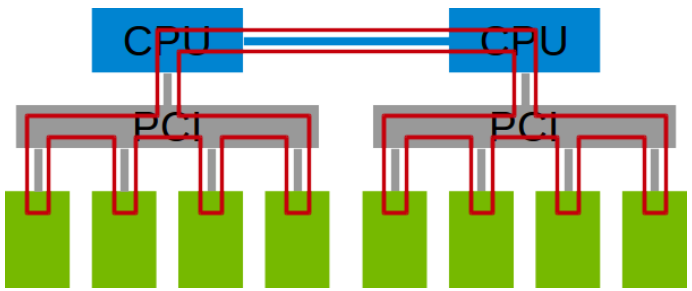
BVLC Caffe – We implemented a tree based reduction in 2014. Good perf on 2 GPUs and DevBox1 shipped with implementation tuned for 4. Didn’t scale well beyond 4 Maxwells

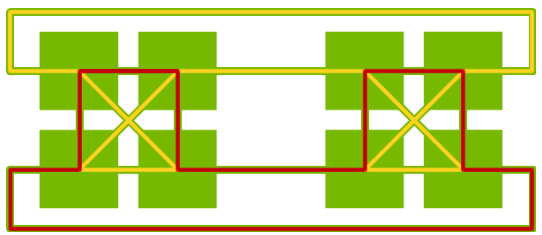
NV-Caffe – NCCL integration, basis for DGX1 performance

Now in every major framework

DATA PARALLEL AND NCCL

NCCL uses rings to move data across all GPUs and perform reductions.
Ring Allreduce is bandwidth optimal and adapts to many topologies

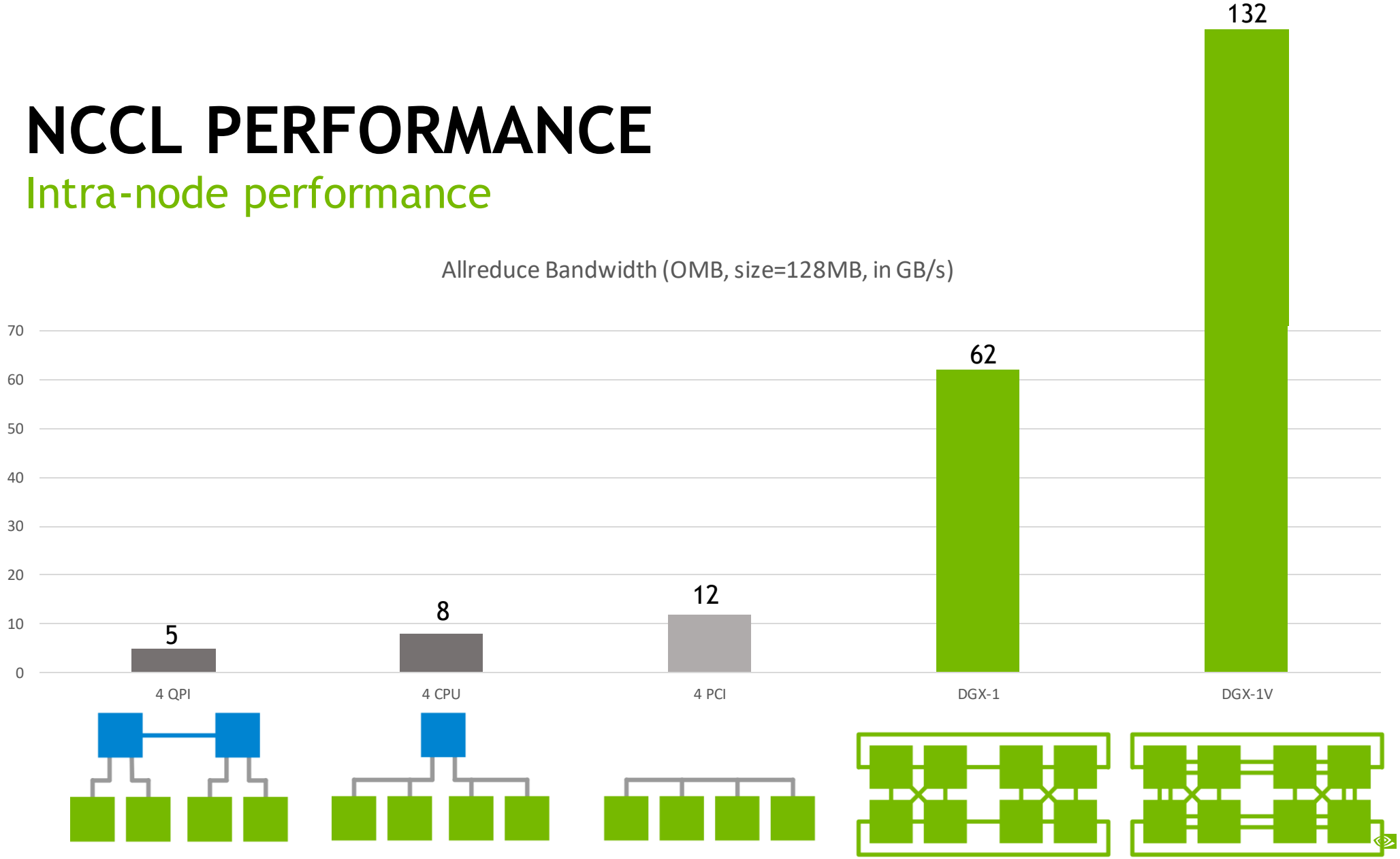




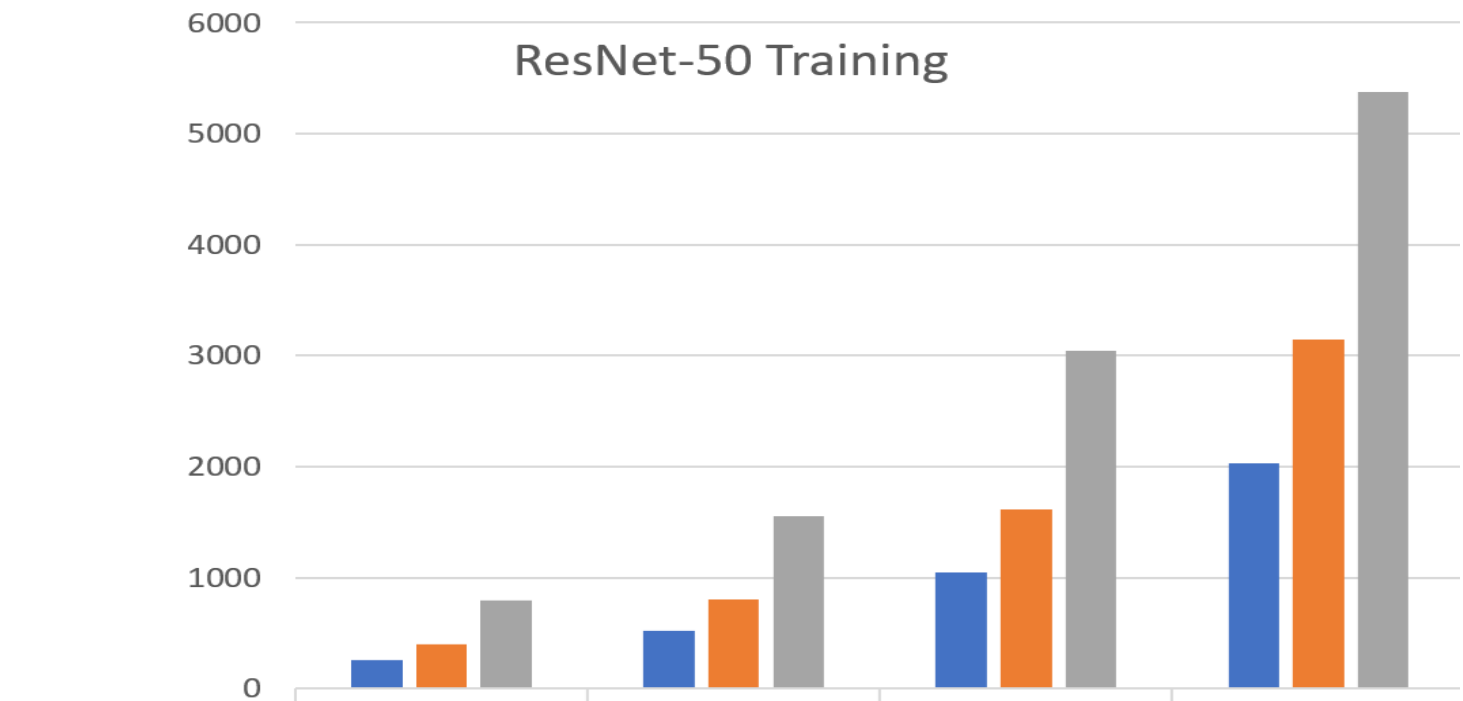
DGX-1 : 4 unidirectional rings

NCCL PERFORMANCE

Intra-node performance



DATA PARALLEL AND LARGE BATCH SIZES



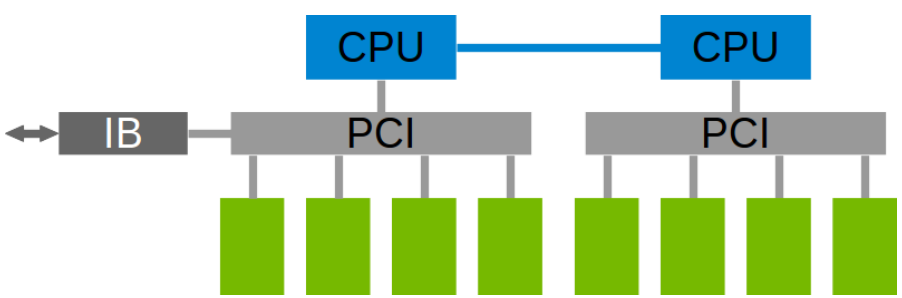
	1 GPU	2 GPU	4 GPU	8 GPU
DGX-1 FP32	261.46	521.04	1045.82	2027.78
DGX-1V FP32	402.13	804.96	1610.25	3142.89
DGX-1V FP16	795.33	1551.33	3040.33	5377.14

NCCL 2.0

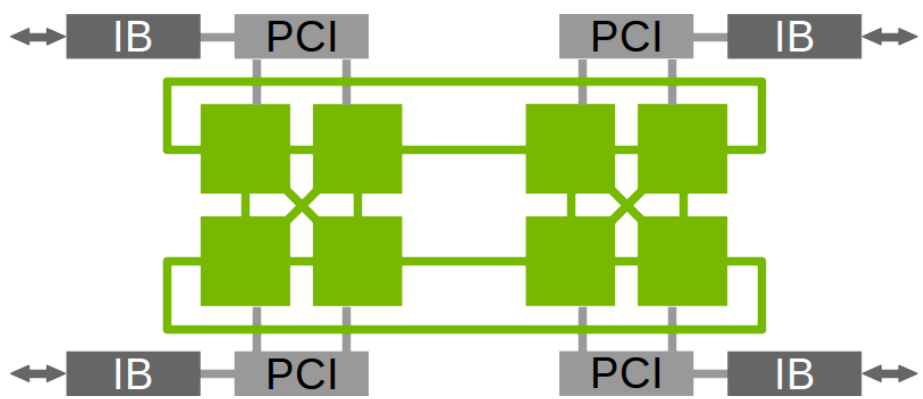
Inter-node communication

Inter-node communication using Sockets or Infiniband verbs, with multi-rail support, topology detection and automatic use of GPU Direct RDMA.

Optimal combination of NVLink, PCI and network interfaces to maximize bandwidth and create rings across nodes.



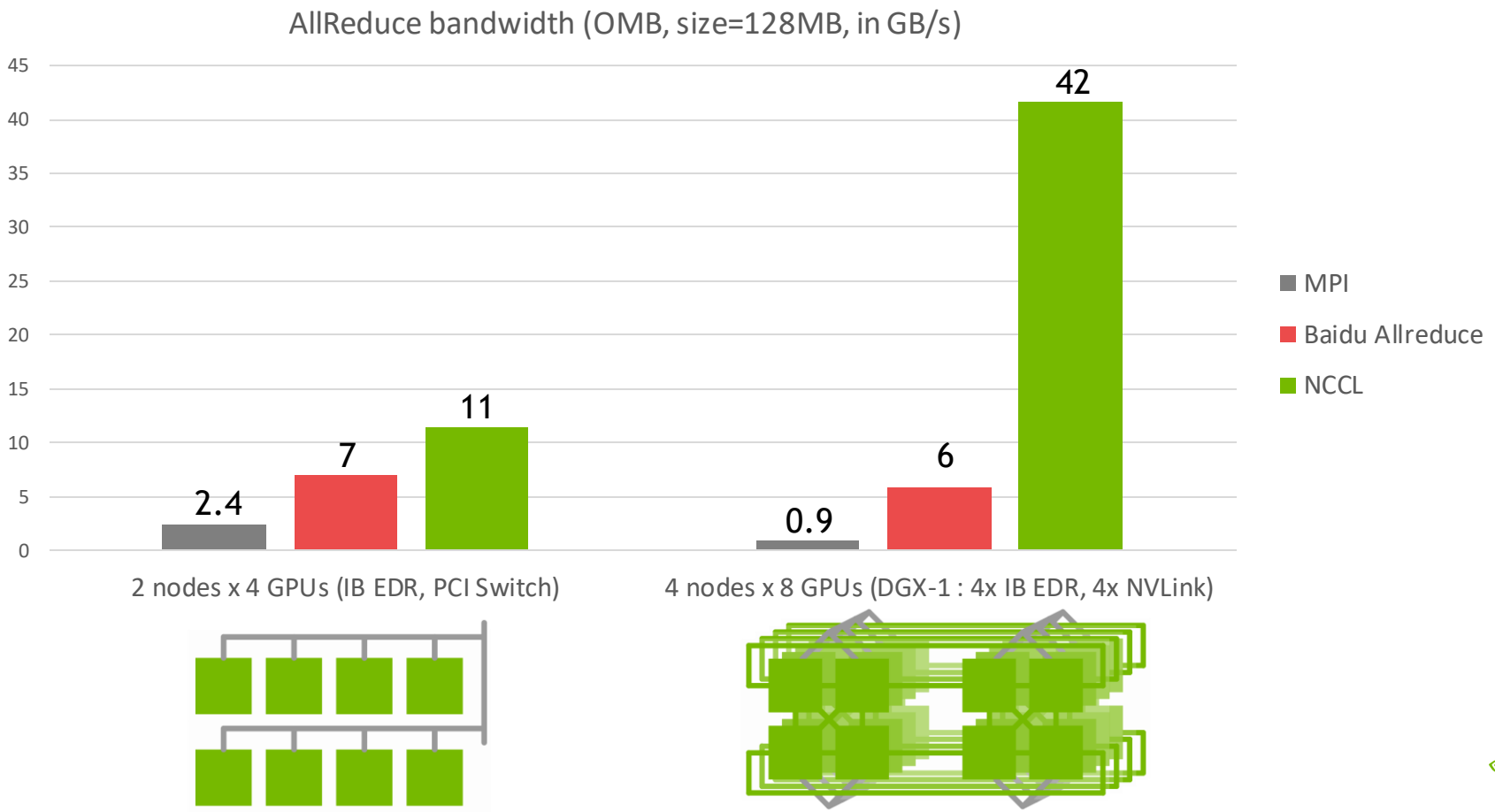
PCIe, Infiniband



DGX-1 : NVLink, 4x Infiniband

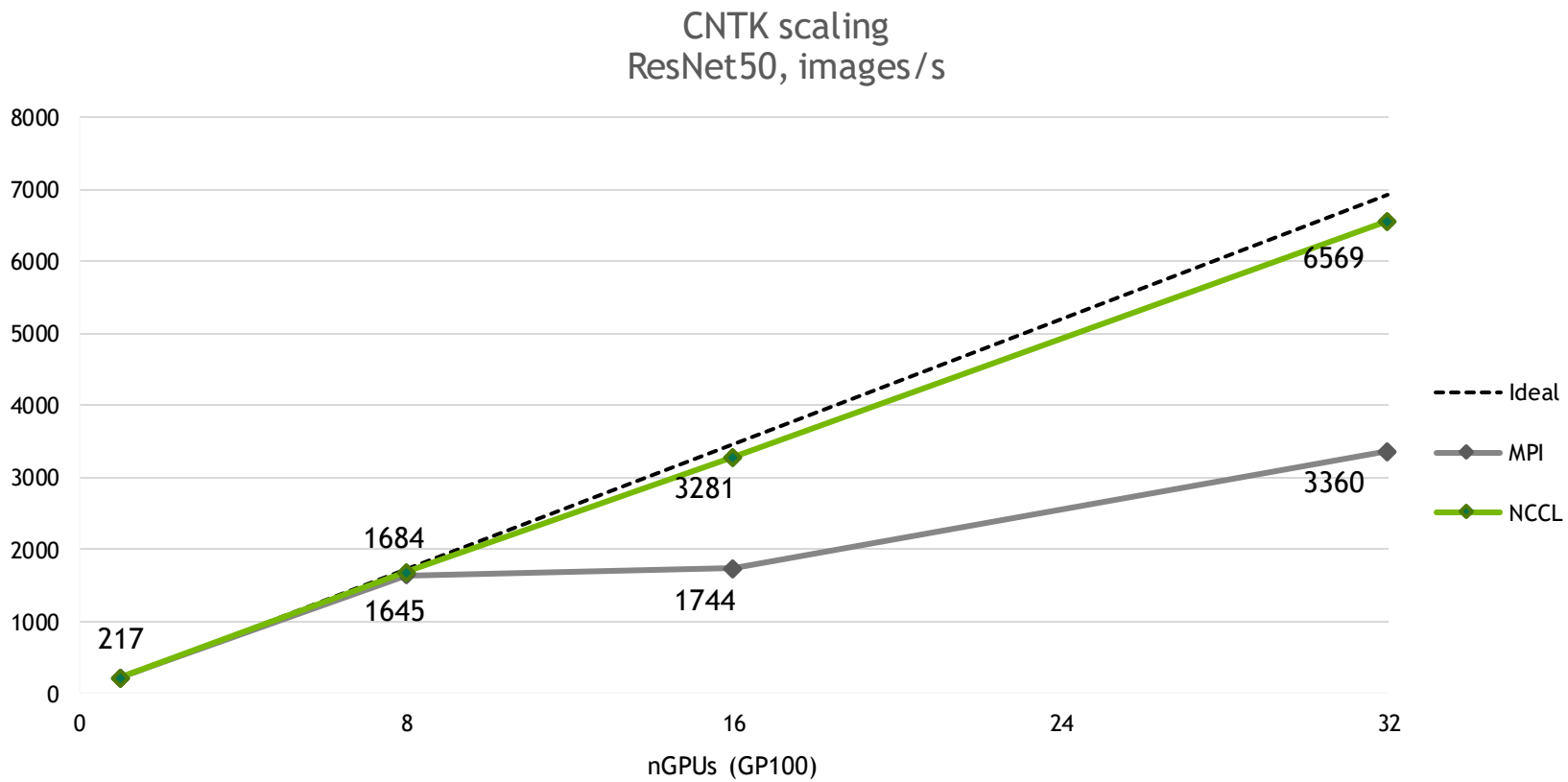
PERFORMANCE

Inter-node performance



PERFORMANCE

Deep Learning - CNTK



NUMERICS ARE HARD WITH LARGE BATCH

DIFFICULTIES OF LARGE-BATCH TRAINING

It’s difficult to keep the test accuracy, while increasing batch size.

Current recipe[Goyal, 2017]:

a linear scaling of learning rate γ as a function of batch size *B*

a learning rate “warm-up” to prevent divergence during initial training phase

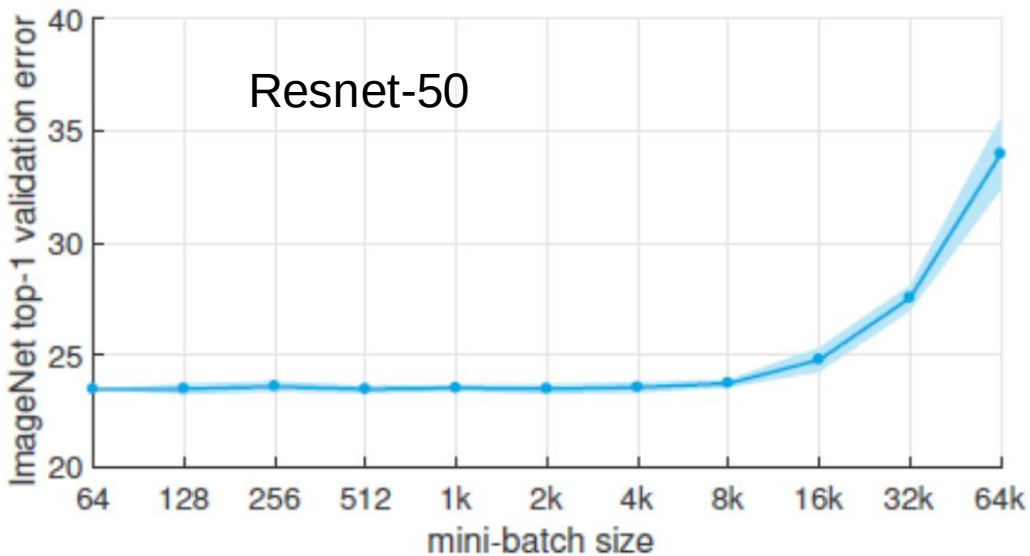
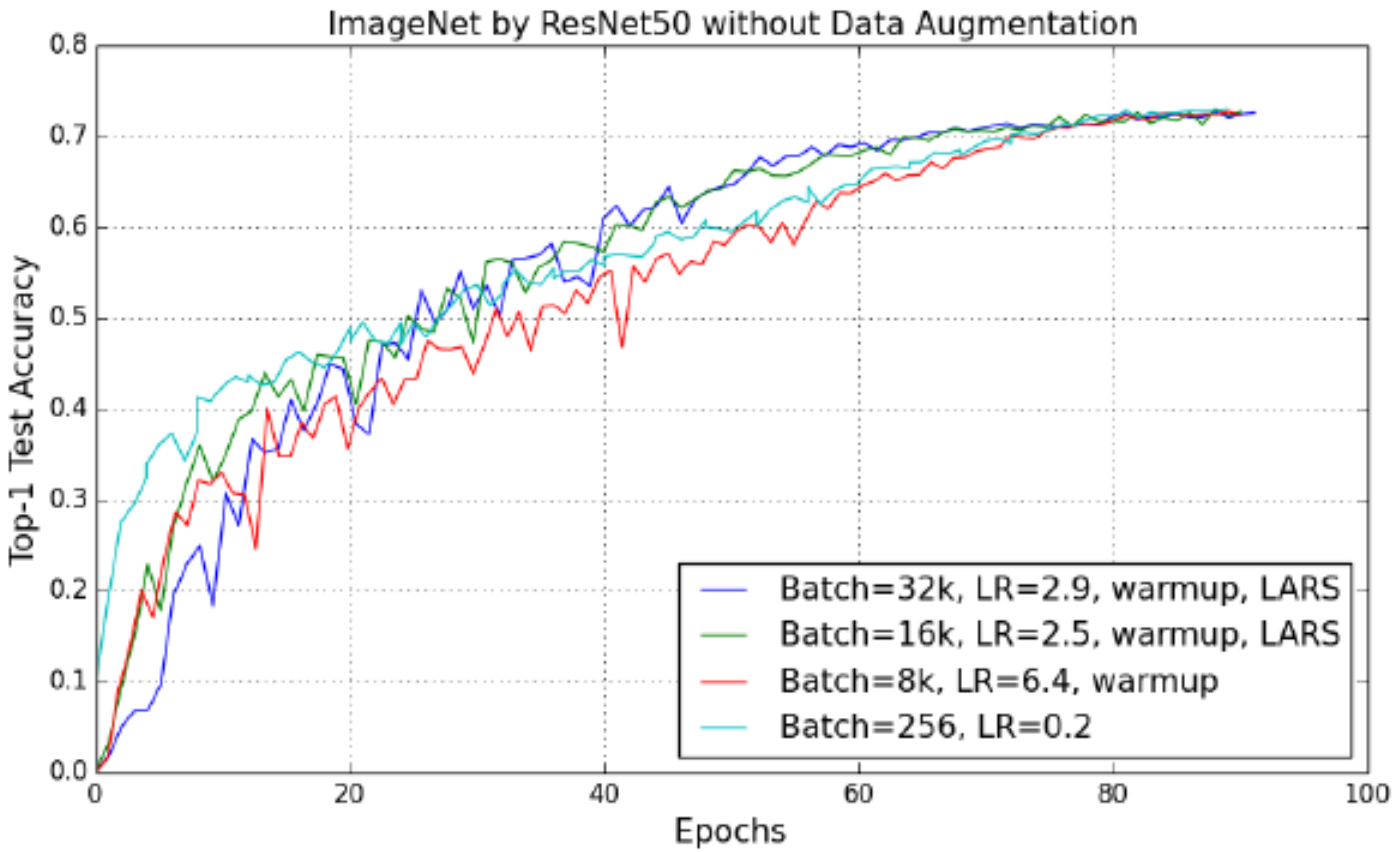


Figure 1. **ImageNet top-1 validation error *vs.* minibatch size.**

Optimization is not a problem if you get right hyper-parameters

Priya Goyal, Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, 2017

RESNET-50 WITH LARS: B→ 32K



More details on LARS in our paper: https://arxiv.org/abs/1708.03888

SUMMARY

1) The key difficulties in large batch training is numerical optimization
2) The existing approach, based on using large learning rates, can lead to divergence, especially during the initial phase, even with warm-up
3) With “Layer-wise Adaptive Rate Scaling”(LARS) we scaled up Resnet-50 to B=16K
4) Even with LARS and warm-up we couldn’t increase LR farther for very large batches. To keep the accuracy we have to increase the number of epochs
5) For Volta efficiency we currently need 128 batch per GPU = 64-128 node limit

ASYNCHRONOUS SGD – A BRIEF TANGENT

Generally Data Parallel

Frequently uses a parameter server

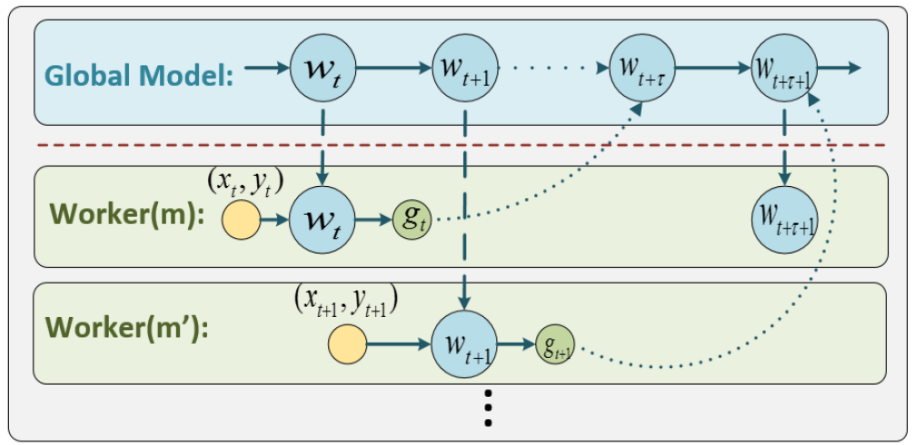
Can update out of phase

Complex numerical behavior

Non-reproducible results

Large reduction in required bandwidth

Somewhat common inside some CSPs



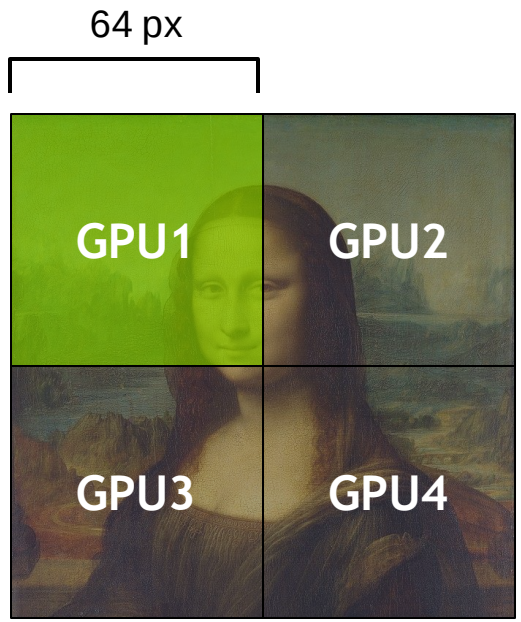
MODEL PARALLEL

MODEL PARALLELISM

Domain decomposition in HPC speak

Suppose the “layer” is 128x128.

If we spatially partition onto 4 GPUs, then each GPU gets a 64x64 spatial “chunk” of the layer.

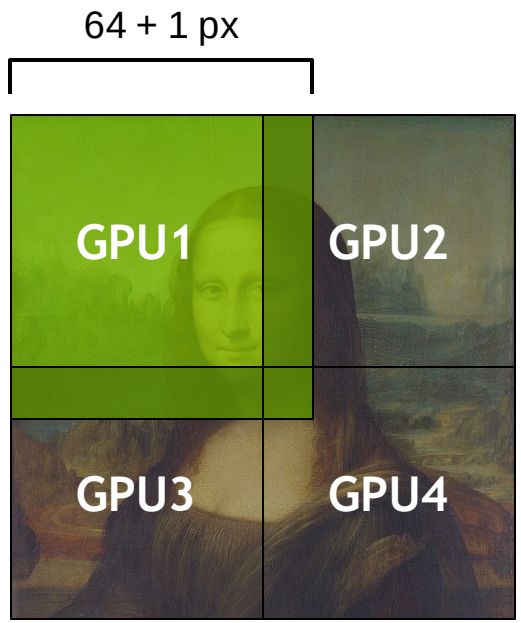


MODEL PARALLELISM

However, if on each GPU we evaluate a 3x3 convolution, we also need to access data from adjacent spatial chunks.

For example, one 3x3 conv with stride 1 has an effective receptive field width of +1 pixel on each side.

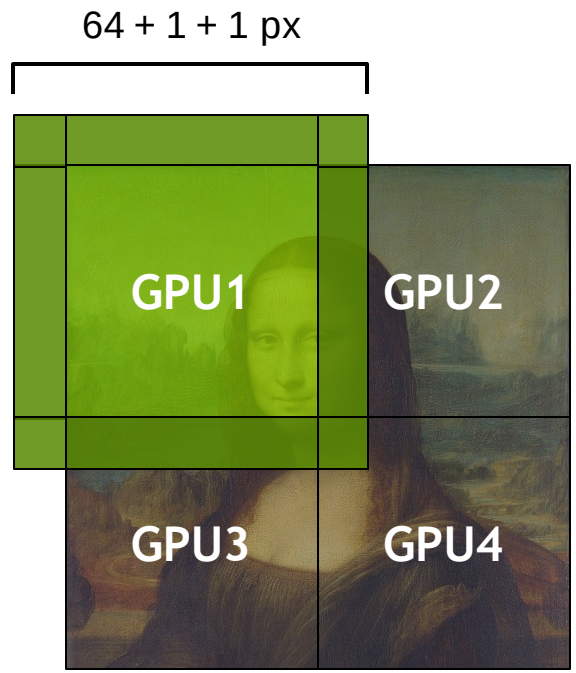
So somehow each GPU needs to be able to see data from adjacent spatial chunks on the other GPUs.



MULTI-GPU MEMORY ACCESS

The prototype implementation of the spatial exchange on current HW consists of a few steps:

1. Define a multi-GPU tensor representation
2. Replace packed single-GPU tensors with padded, spatially partitioned multi-GPU tensors
3. Exchange boundary data between spatially adjacent padded tensors using an explicit communication kernel
4. Use the padded tensors as explicitly padded inputs to convolutions



SUMMARY

We can split layers across GPUs in several ways

Allows for larger models if we can treat as “GPU SMP’

Scaling at 16 GPUs looks good, but worried about having enough work per GPU at scale with a ResNet-50 like network

Working on improvements to memory model over NVLink

HOW ABOUT A BIT OF BOTH?

HOW TO GO REALLY BIG?

OR I HAVE $100M TO DEVOTE TO A SINGLE ENGINEER/PROJECT

ENSEMBLE/MOE TRAINING

Common in massive scale training

Train many models at once in search of design and hyperparameter space

- Hyperparam space is O(100) variables
- Network design space is O(1000) variables

"Learning to Learn" - https://arxiv.org/pdf/1606.04474.pdf

Train lots of different models

- Ensemble
- Mixture of Experts

"Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer" - https://arxiv.org/pdf/1606.04474.pdf

SUMMARY

Where we need help

- We have several potential paths for scaling
  - Good evidence CNNs will work – see NERSC Gordon Bell Submission
  - RNNs look harder as they don't like large batch and LARS doesn't work, yet
  - Implementation across all frameworks a huge lift
- Infrastructure
  - Need to clean up container support for Infiniband to work nicely
  - Failover support
  - Really need a new type of IO hierarchy
- SW
  - Multi-node model parallel techniques
  - Solver/numerical studies and support for massive scale