In-Network Computing to Enable Large Scale Deep Learning Training

1.0

Outline

- Why do we need distributed training ?
- Distributed training models
- Mellanox Scalable Hierarchical Aggregation and Reduction Protocol (SHARP)
- Performance measurements

Why Do We Need Distributed Training ?

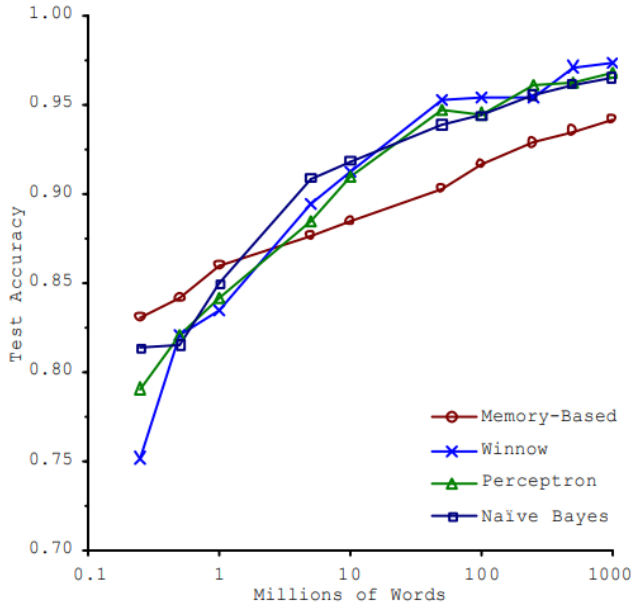
We want our networks to have **better accuracy** and provide more **complex results**

Need to extract more features, remember longer sequences and increase the model output size

-> increase in compute intensity and model size

Why Do We Need Distributed Training ?

There is no data like more data



[Banko and Brill,2001]

Why Do We Need Distributed Training ?

A glance on MLPerf HPC possible benchmarks

More data with increase in complexity

Benchmark	Dataset	Relevant Performance Characteristics for HPC
ResNet 101	TenCent Images **2.2 TB**	• Larger than Imagenet • Model utilizes multi-label loss • **Stresses network bandwidth more than ResNet50**
Translation	WMT English-to-French **2.5 GB**	• Model Parallelism required for accuracy improvements • Multi-node scaling stresses network latency and bandwidths • **Communication patterns unique to sequence model-parallelism**
Climate Segmentation	Community Atmosphere Model Simulation output **20 TB**	• High-dimensional inputs and multi-label targets • 16-channel 2D images of size (768, 1152, 16), 3-class per-pixel • **Large dataset (20 TB) stresses I/O subsystem**
Cosmoflow	PyCola N-Body Simulation output **40 TB**	• Hybrid Data-Parallel + Model-Parallel across 3D Convolutions • **Multi-channel Volumetric input data stresses on-node bandwidths and memory capacities** • **Communication patterns unique to convolutional model-parallelism**

[MLPerf HPC]

Why Do We Need Distributed Training ?

Training is an intensive task

model complexity + large data set

=

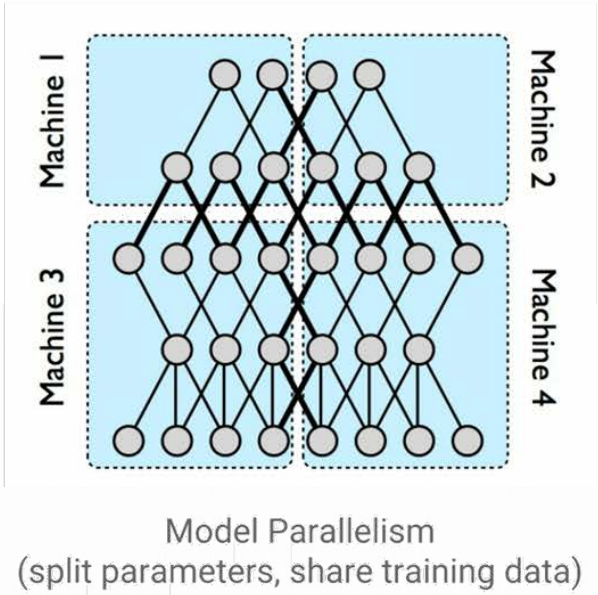
increase training time

MLPerf v0.5 Results – https://mlperf.org/results/

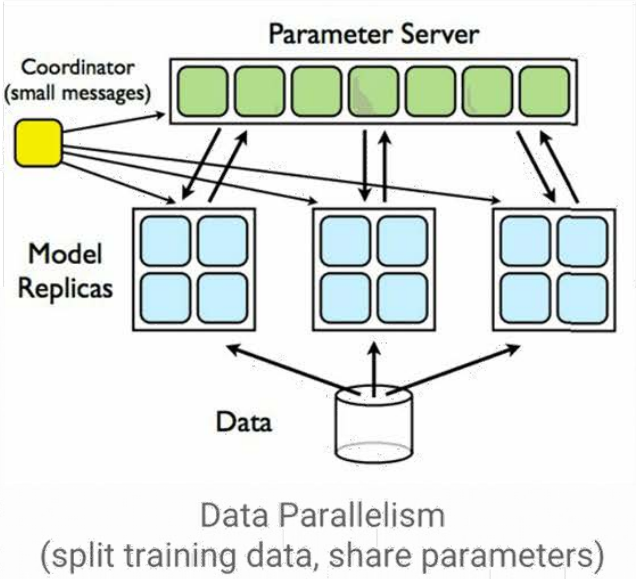
Un-optimized reference: single pascal will need ~6Days to train Resnet50 with ImageNet

Distributed Training Models

Basic models – model/data parallelism (or both)



Model Parallelism
(split parameters, share training data)

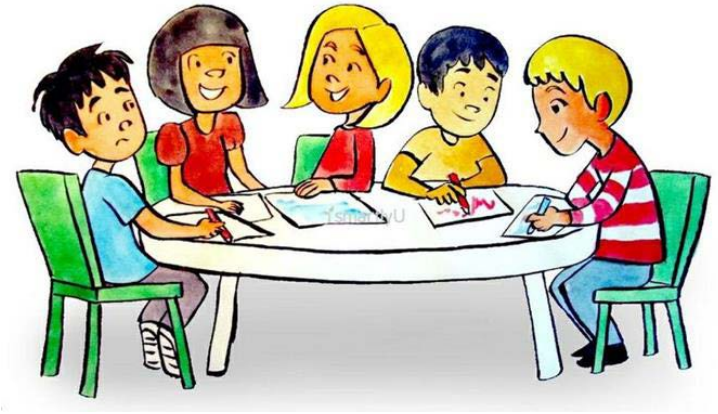


Data Parallelism
(split training data, share parameters)

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/40565.pdf

Distributed Training Models

- Data parallelism requires synchronization between the models – calculating the average change per parameter
- As models are getting bigger there are more parameters to sync



Distributed Training Models

- Calculating the average change is basically an “All-Reduce operation”
  - AllReduce completion time **increase** with vector size and number of models
- In order to reduce this impact people are exploring other ideas –

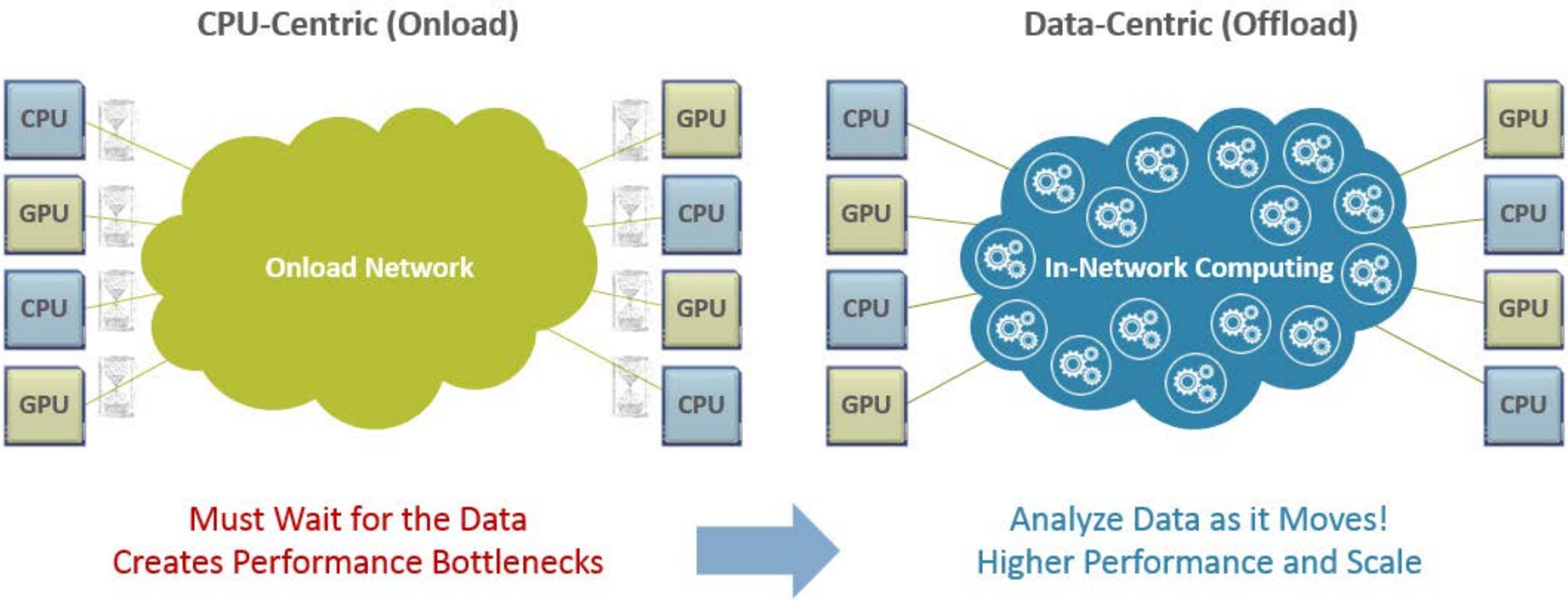
  complex algorithms for partial parameters updates

  reduced precision

  both may require special hardware on the main accelerator or and additional software overhead

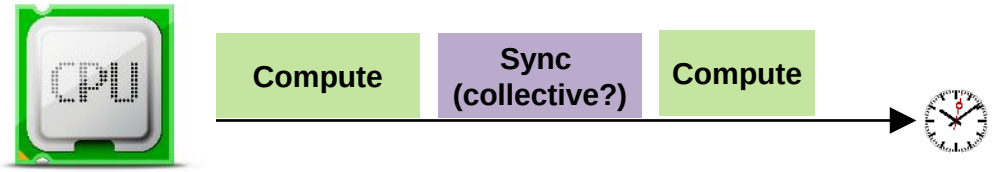
Mellanox's Solution

Intelligent and Faster Interconnect

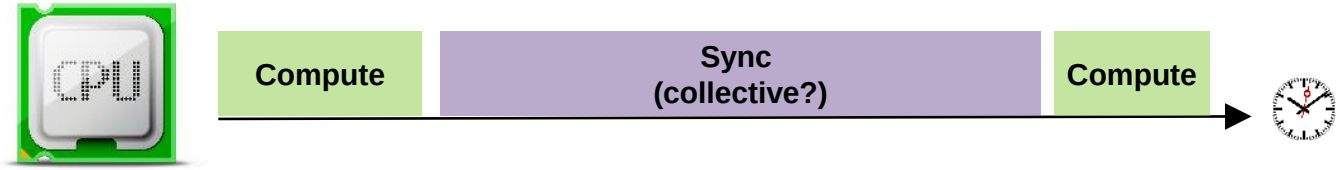


How Can In-Network Computing Help?

Synchronization and collective can become a bottleneck

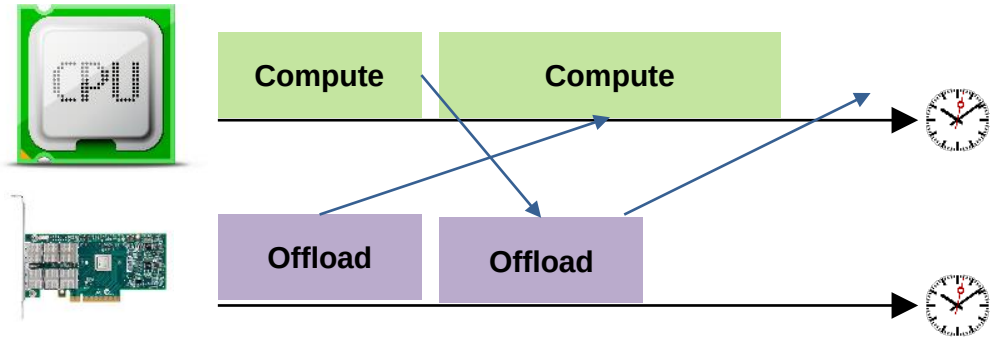


When scale and operation size increase the sync stage latency will increase



How Can In-Network Computing Help?

Network offloads accelerate the operations and let the compute handle their own business



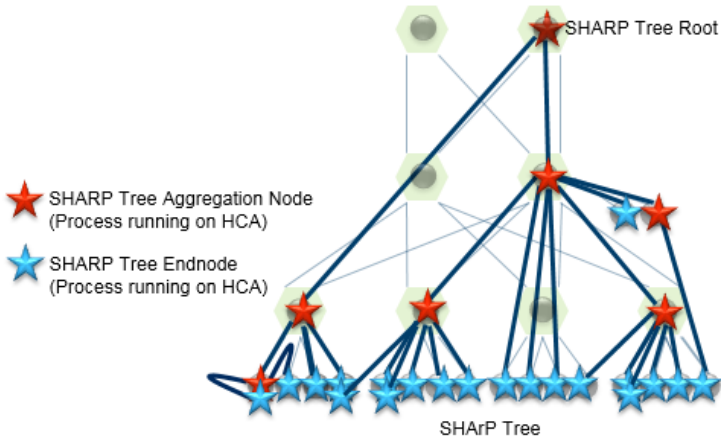
Scalable Hierarchical Aggregation and Reduction Protocol (SHARP)

- Reliable Scalable General Purpose Primitive
  - In-network Tree based aggregation mechanism
  - Large number of groups
  - Multiple simultaneous outstanding operations

SHARP
Scalable Hierarchical Aggregation and Reduction Protocol

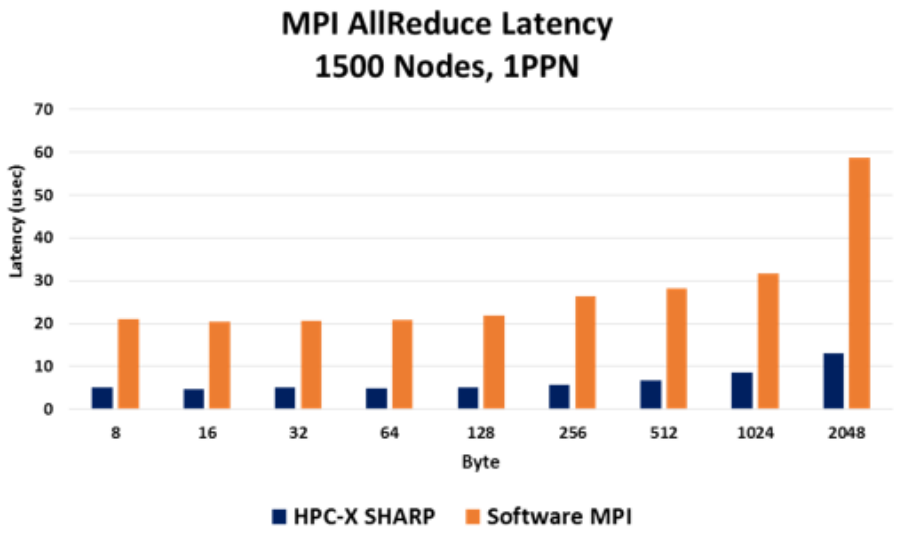
- Applicable to Multiple Use-cases
  - HPC Applications using MPI / SHMEM
  - Distributed Machine Learning applications

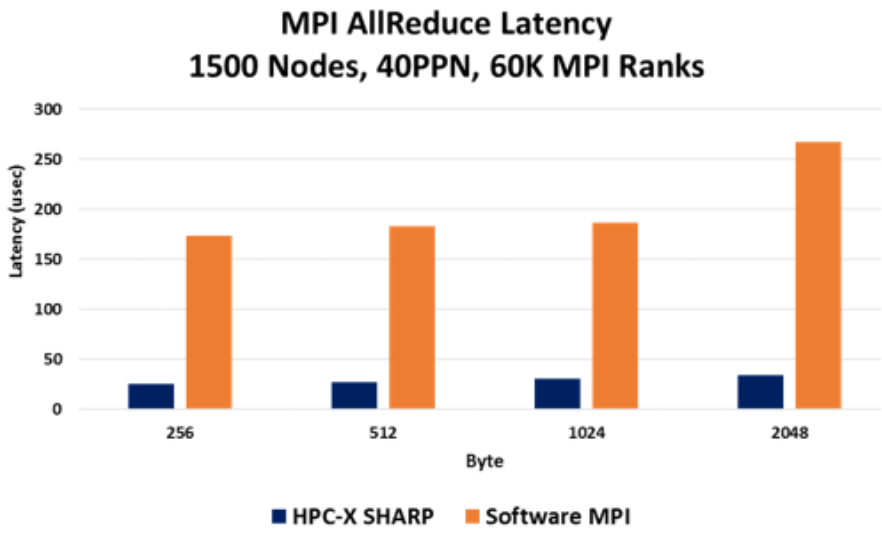
- Scalable High Performance Collective Offload
  - Barrier, Reduce, All-Reduce, Broadcast and more
  - Sum, Min, Max, Min-loc, max-loc, OR, XOR, AND
  - Integer and Floating-Point, 16/32/64 bits



SHARP All-Reduce Performance Advantages

1500 Nodes, 60K MPI Ranks, Dragonfly+ Topology

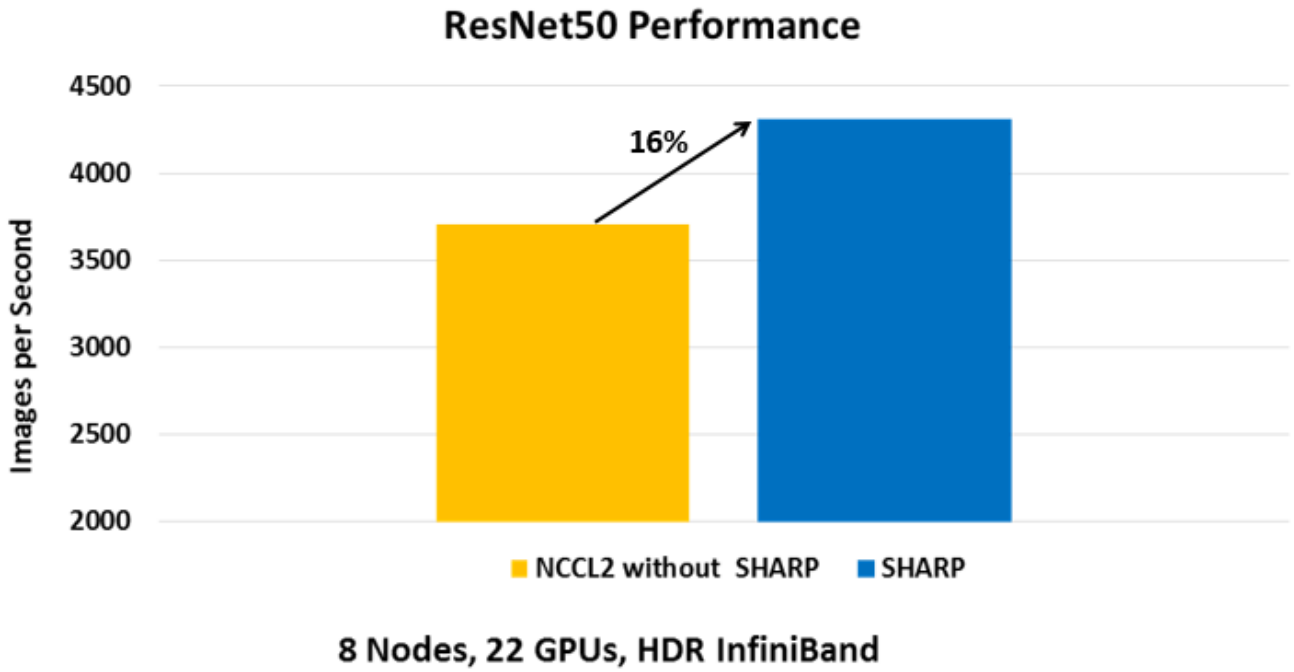




*Lower is better

SHARP Performance Advantage for AI

SHARP provides 16% Performance Increase for deep learning, initial results
TensorFlow with Horovod running ResNet50 benchmark, HDR InfiniBand (ConnectX-6, Quantum)



Mellanox TECHNOLOGIES

YEAR ANNIVERSARY

Read more about Mellanox – http://www.mellanox.com/

Mellanox HDR 200G InfiniBand Deep Learning Acceleration Engines Demonstrates Two Times Higher Performance for Artificial Intelligence (AI) Platforms with NVIDIA

Mellanox In-Network Computing "Hierarchical Aggregation and Reduction Protocol" (SHARP)™ Technology in Combination with NVIDIA Collective Communications Library (NCCL) Delivers Performance Breakthrough to AI

https://www.businesswire.com/news/home/20190318005282/en/

Takeaways

If you want to train real models faster -> distributed training

Data parallelism requires all reduce operations (eliminating parameter server)

AllReduce latency increase when model size increase and distribution scale increase

Mellanox's SHARP can offload AllReduce operations and reduce the operation latency -> accelerate distributed training

Questions ?