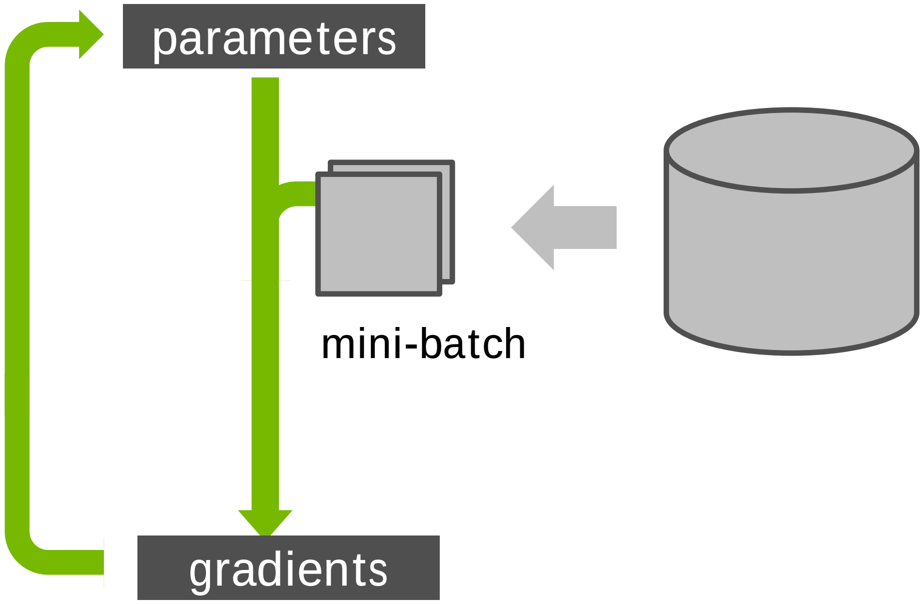
PRACTICAL SCALING TECHNIQUES

Ujval Kapasi | Dec 9, 2017

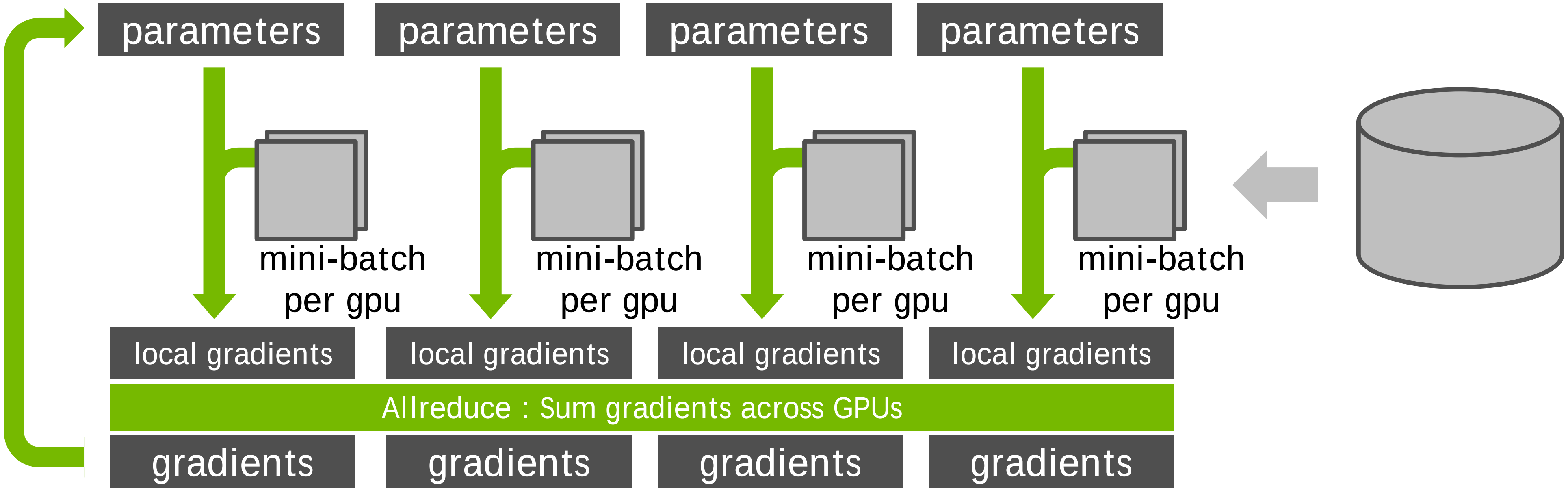
DNN TRAINING ON MULTIPLE GPUS

Making DL training times shorter



DNN TRAINING ON MULTIPLE GPUS

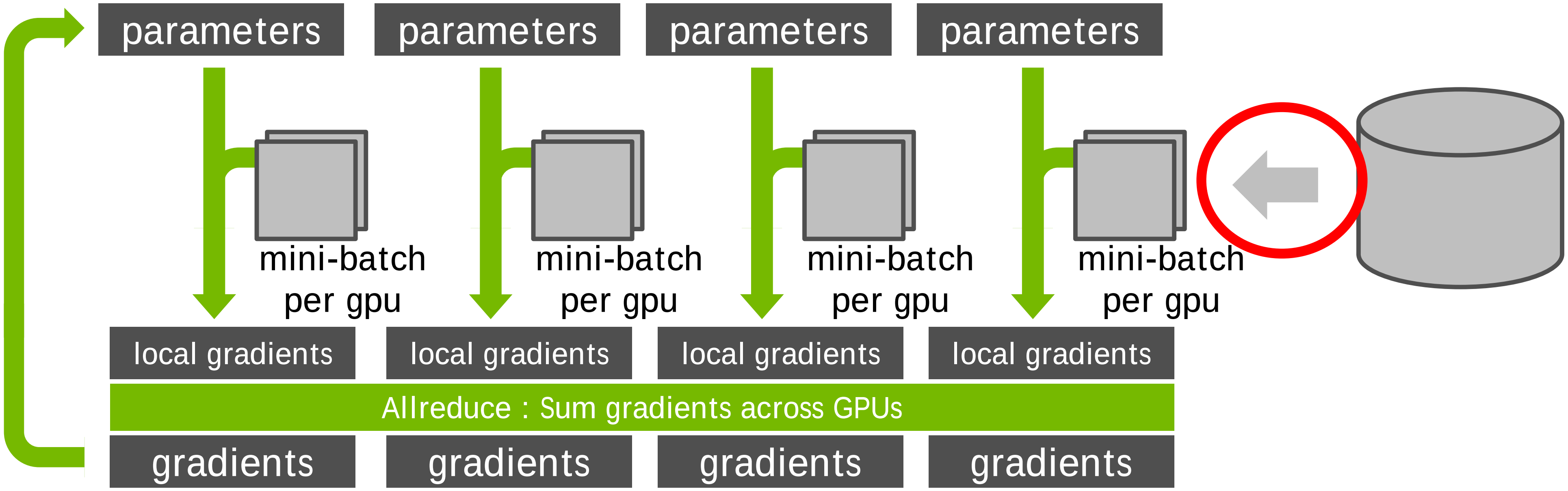
Making DL training times shorter



Data parallelism : split batch across multiple GPUs

DNN TRAINING ON MULTIPLE GPUS

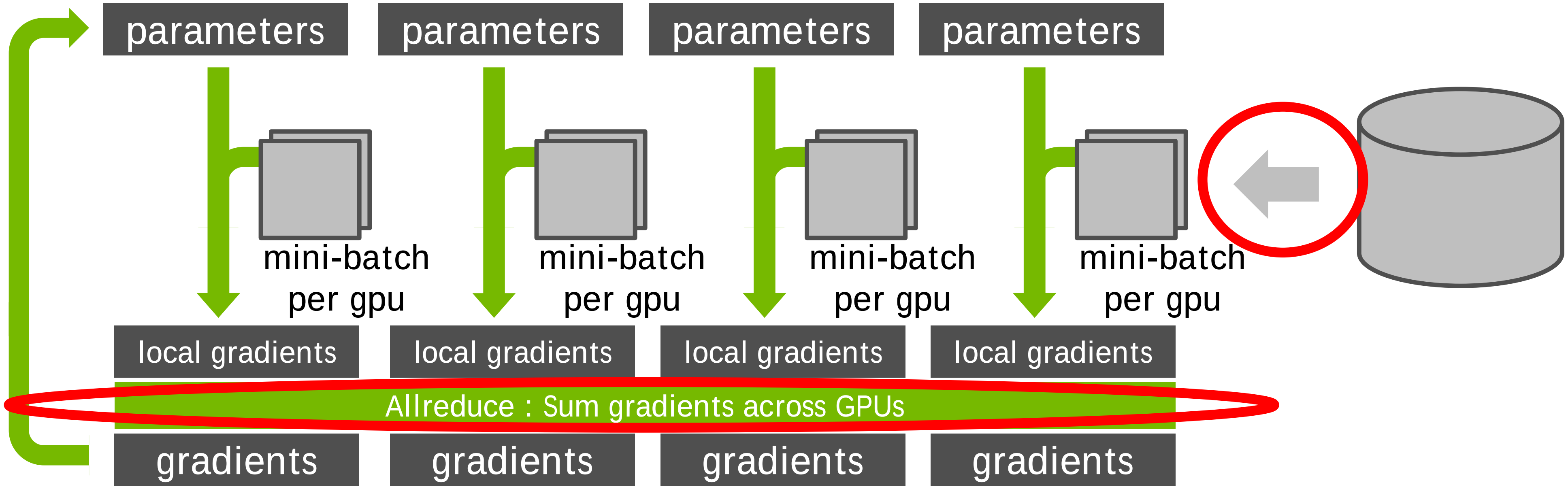
Making DL training times shorter



Data parallelism : split batch across multiple GPUs

DNN TRAINING ON MULTIPLE GPUS

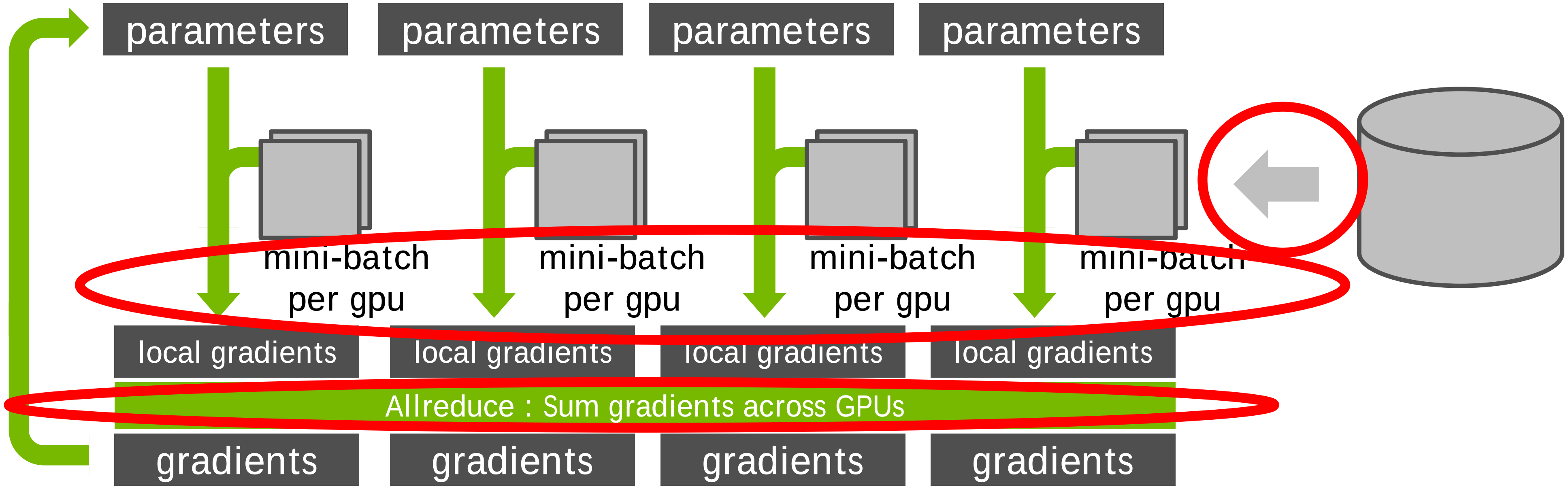
Making DL training times shorter



Data parallelism : split batch across multiple GPUs

DNN TRAINING ON MULTIPLE GPUS

Making DL training times shorter

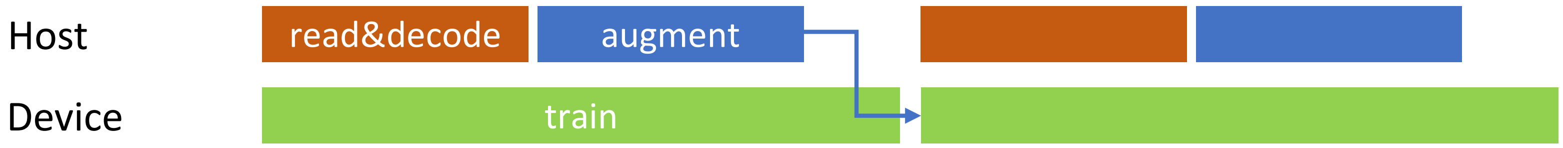


Data parallelism : split batch across multiple GPUs

OPTIMIZE THE INPUT PIPELINE

TYPICAL TRAINING PIPELINE

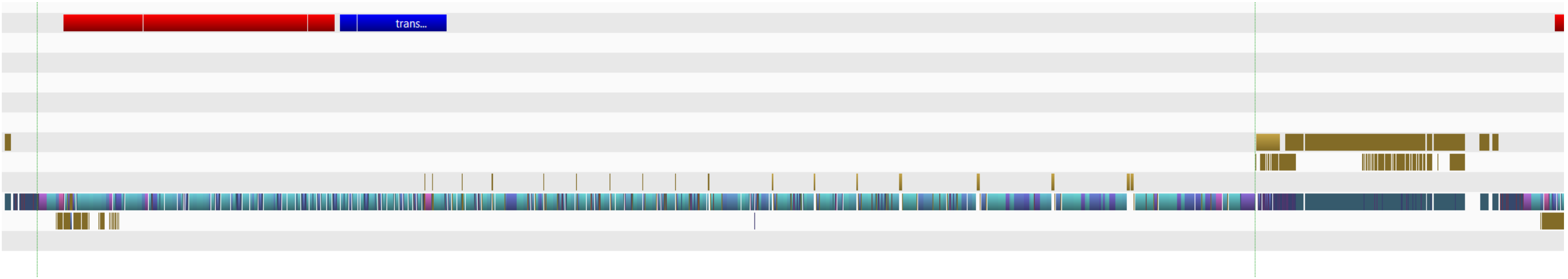
- Device/Training limited



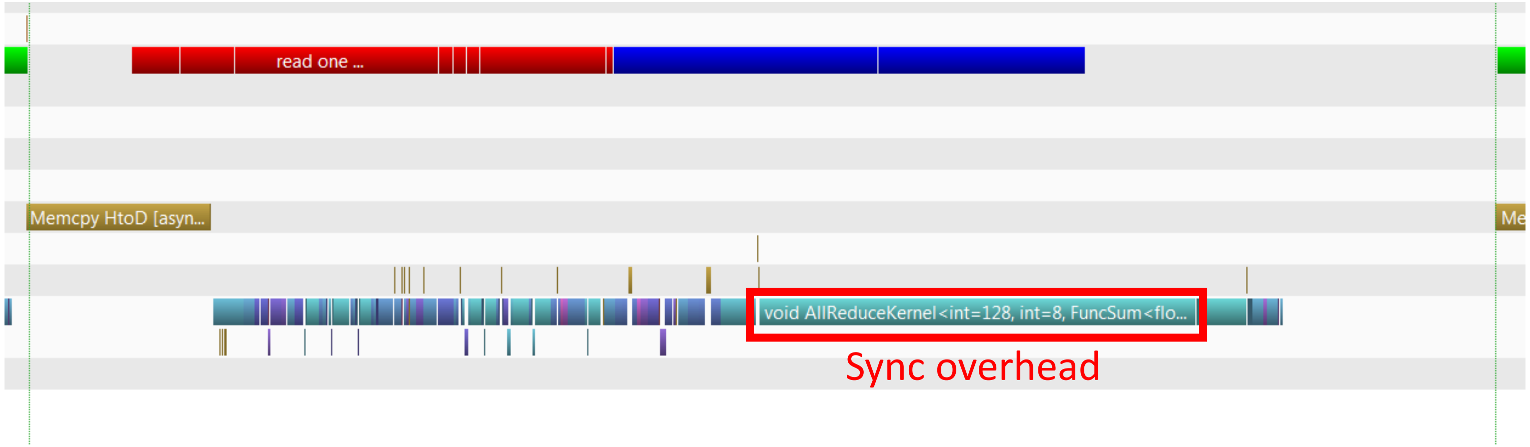
- Host/IO limited

EXAMPLE: CNTK ON DGX-1

- Device/Training limited (ResNet 50)



- Host/IO limited (AlexNet)



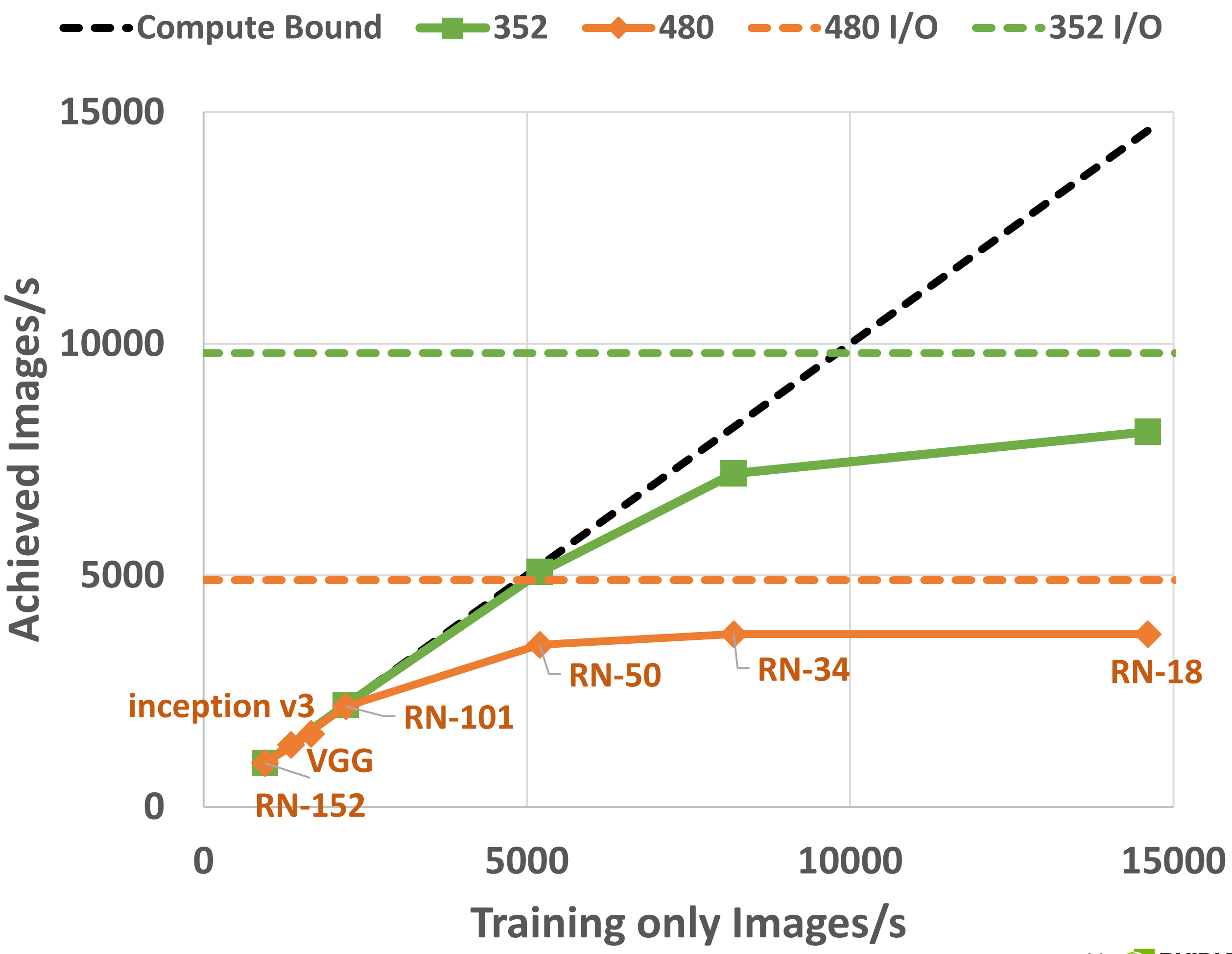
THROUGHPUT COMPARISON

	IMAGES/SECOND	
File I/O	~10,000	290x550 images ~1600 MB/s with LMDB on DGX-1 V100
Image Decoding	~ 10,000 – 15,000	290x550 images libjpeg-turbo, OMP_PROC_BIND=true on DGX-1 V100
Training	>6,000 (Resnet-50) >14,000 (Resnet-18)	Synthetic dataset, DGX-1 V100

ROOFLINE ANALYSIS

Horizontal lines show I/O pipeline throughputs

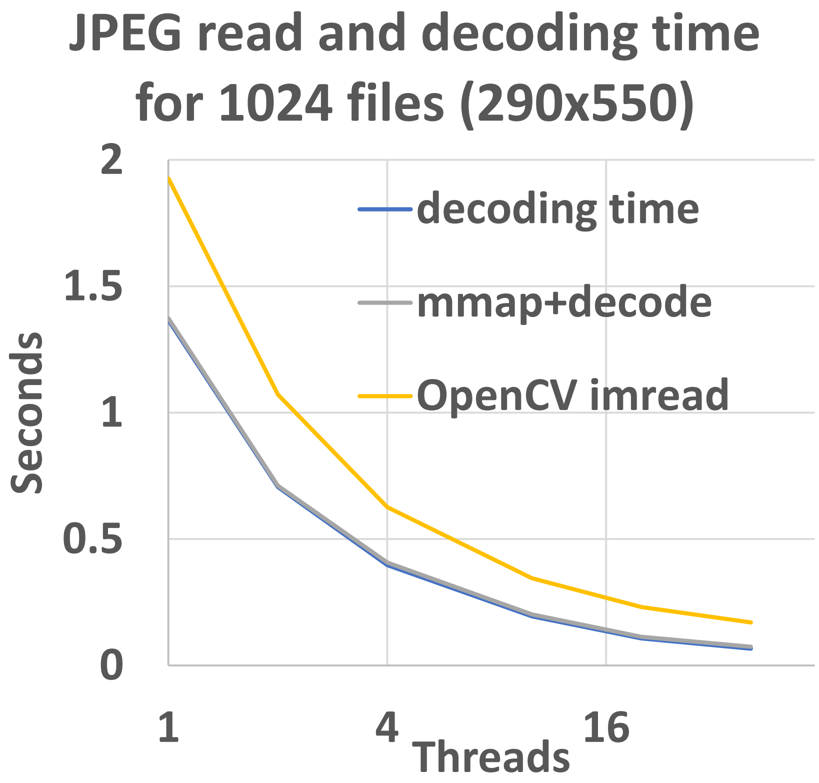
- 352: 9800 images/s
- 480: 4900 images/s



Measurements collected using the Caffe2 framework

RECOMMENDATIONS

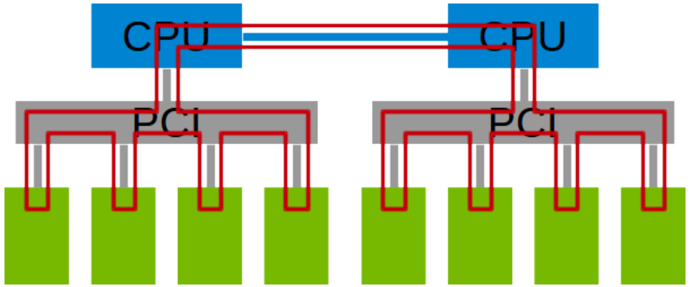
- Optimize Image I/O:
  - Use fast data and file loading mechanisms such as LMDB or RecordIO
  - When loading from files consider mmap instead of fopen/fread
  - Use fast image decoding libraries such as libjpeg-turbo
  - Using OpenCVs imread function relinquishes control over these optimizations and sacrifices performance
- Optimize Augmentation
  - Allow augmentation on GPU for I/O limited networks

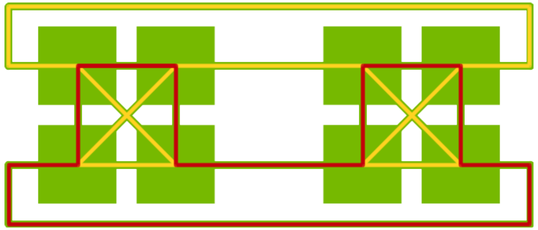


OPTIMIZE COMMUNICATION

DATA PARALLEL AND NCCL

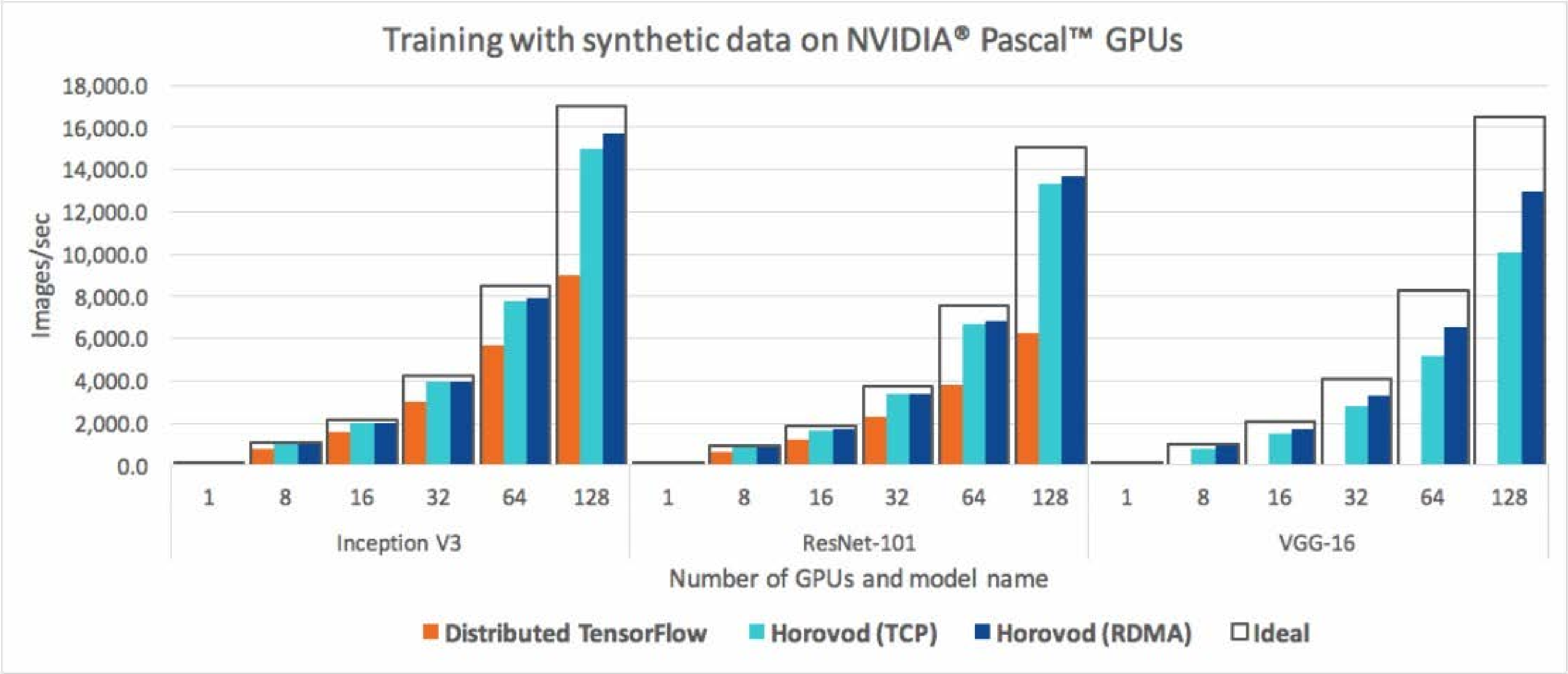
NCCL uses rings to move data across all GPUs and perform reductions.
Ring Allreduce is bandwidth optimal and adapts to many topologies





DGX-1 : 4 unidirectional rings

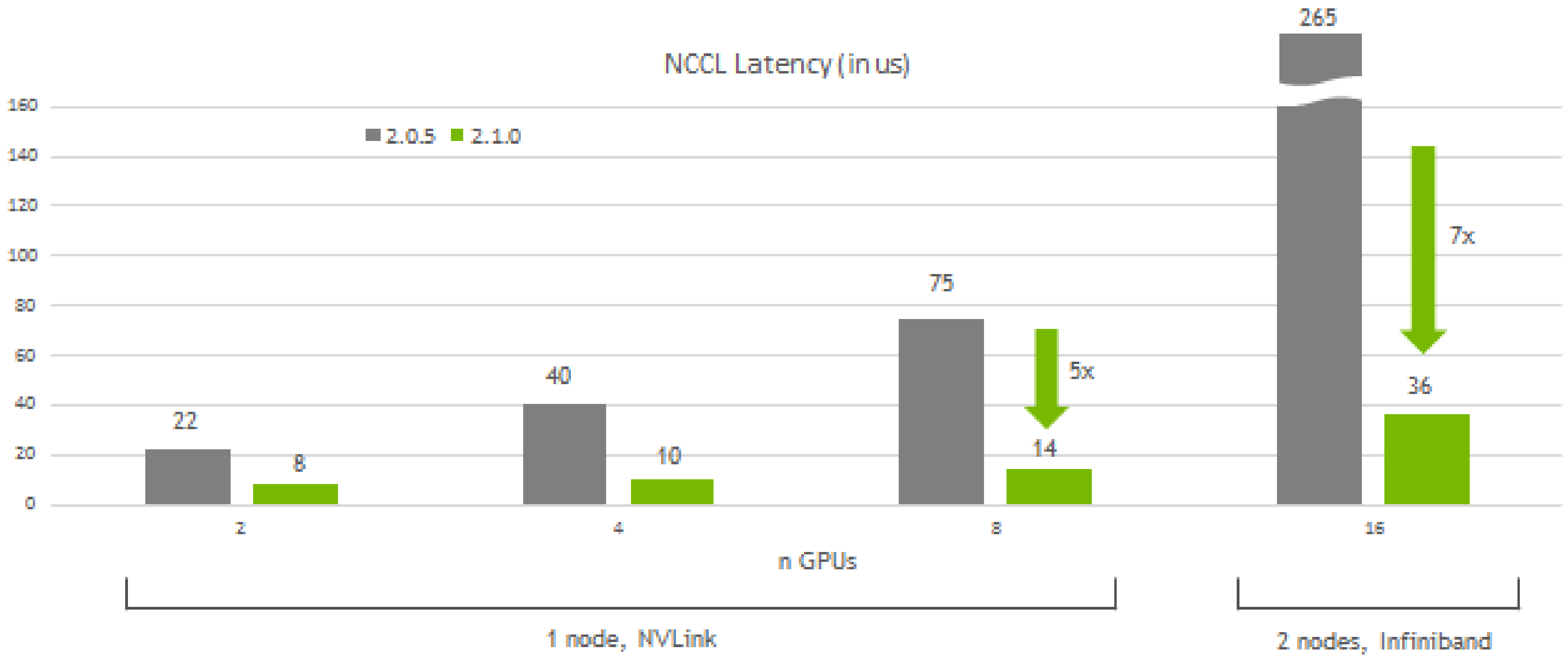
NCCL 2 MULTI-NODE SCALING



https://github.com/uber/horovod

NCCL 2.1

Latency improvement



TRAIN WITH LARGE BATCHES

DIFFICULTIES OF LARGE-BATCH TRAINING

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

Recipe from [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

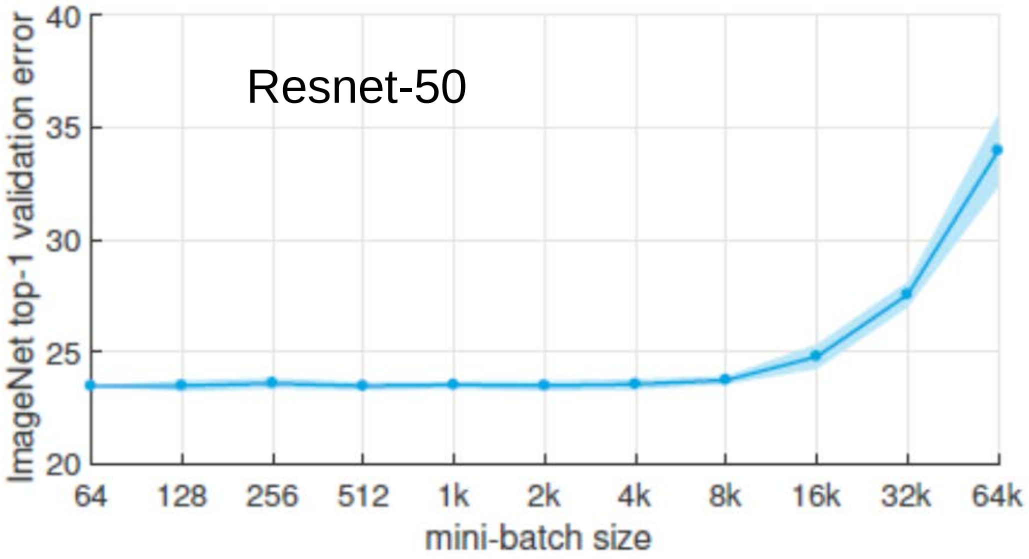


Figure 1. **ImageNet top-1 validation error *vs*. minibatch size.**

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

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

LARGE-BATCH TRAINING

Sam Smith, et al. “Don’t Decay the Learning Rate, Increase the Batchsize”

Keskar, et al. “On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima”

Akiba, et al. “Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes”

LAYER-WISE ADAPTIVE RATE SCALING (LARS)

Use local LR λ^l for each layer l

$$\Delta \boldsymbol{w_t^l} = \boldsymbol{\gamma} * \boldsymbol{\lambda^l} * \boldsymbol{\nabla L(w_t^l)}$$

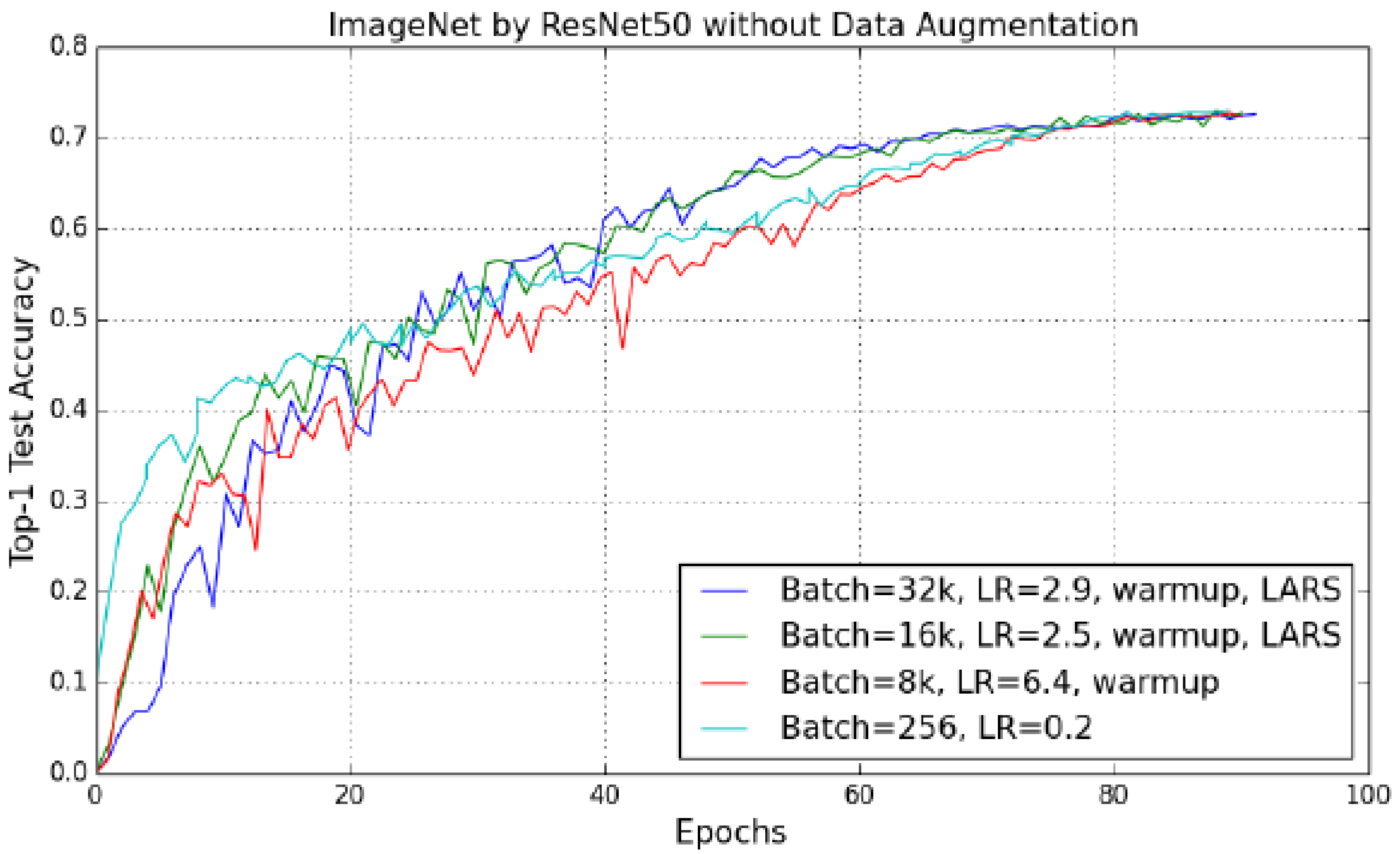
where:

γ – global LR,

$\nabla L(w_t^l)$ – stochastic gradient of loss function L wrt w_t^l

λ^l – local LR for layer l

RESNET-50 WITH LARS: B→ 32K



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

SUMMARY

1) Larger batches allow scaling to larger number of nodes while maintaining high utilization of each GPU

2) The key difficulties in large batch training is numerical optimization

3) The existing approach, based on using large learning rates, can lead to divergence, especially during the initial phase, even with warm-up

4) With “Layer-wise Adaptive Rate Scaling”(LARS) we scaled up Resnet-50 to B=16K

FUTURE CHALLENGES

Hybrid Model parallel and Data parallel

Disk I/O for large datasets that can't fit in system memory or on-node SSDs

NVIDIA