Communication Backend Intro

Zhaobo Zhang

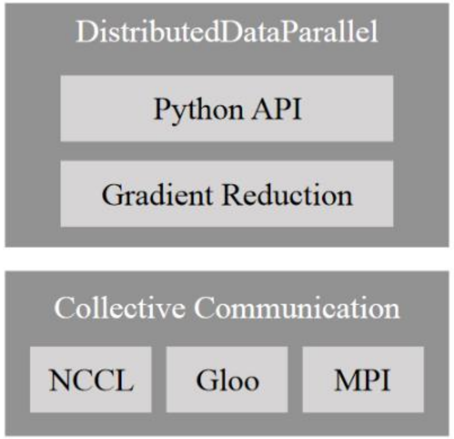
Aug. 4^{th}, 2021

Outline

- Background
- Collective communication
- All-reduce algorithms
- Comparison of NCCL and Gloo
- Usage recommendation

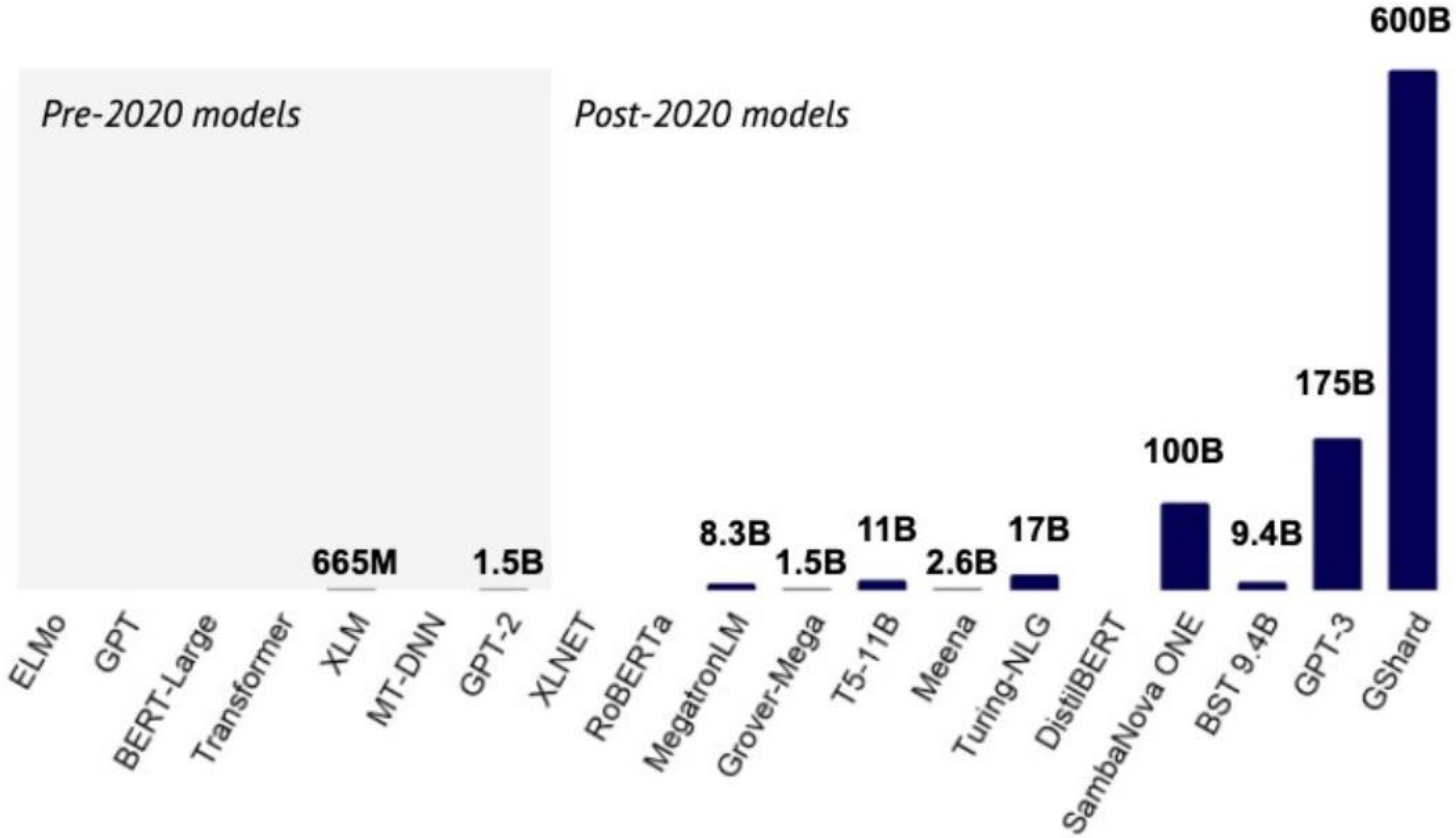
Background

- Distributed training requires sharing learnt gradients among all workers
- Gradient communication during training takes almost 62% of the total execution time among all our workloads on average [1]
- Faster communication reduces training job competition time



Distributed Data Parallel Building Blocks [2]

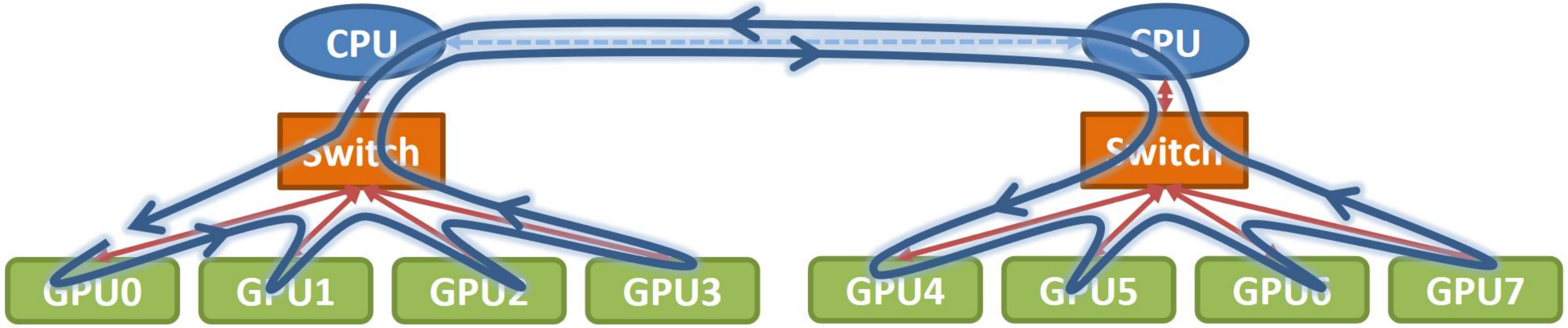
Model size (no. of parameters)



stateof.ai 2020

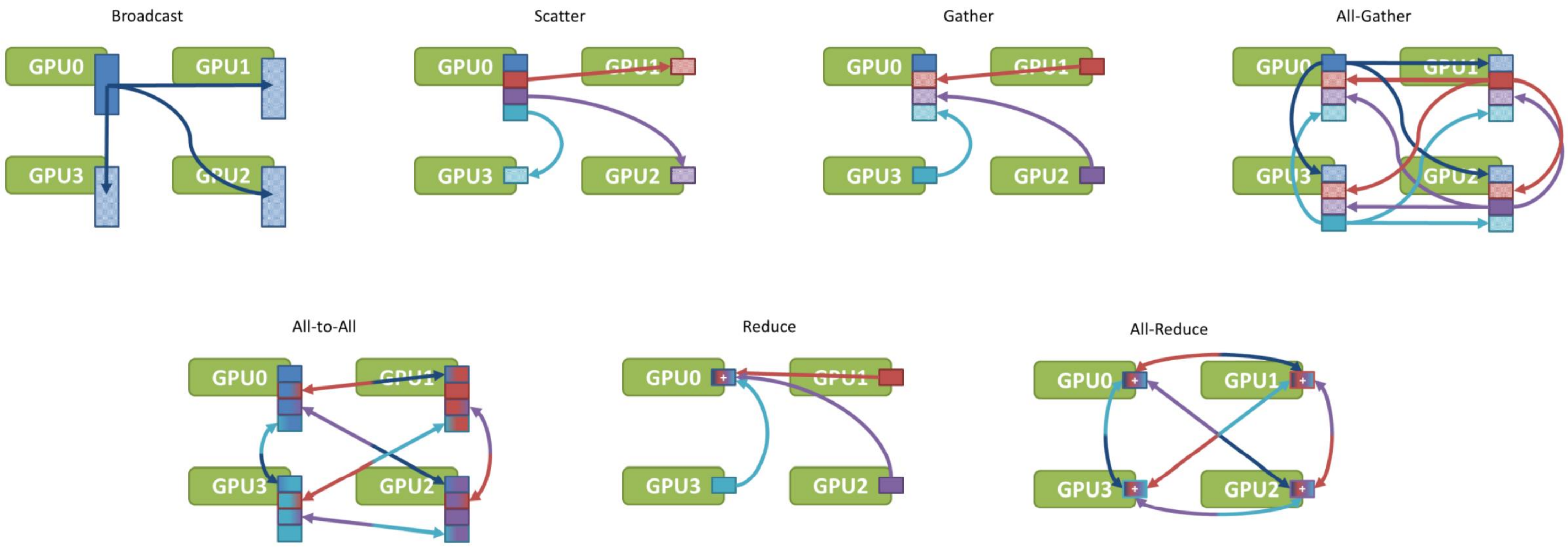
Communication bandwidth

- Inter-node communication
  - Ethernet: 1~100Gbps
  - InfiniBand: QDR (40Gb/s) and FDR (56Gb/s)
- Intra-node communication
  - PCIe gen3x16 (16 GB/s)
  - PCIe gen4x16 (32 GB/s)
  - NVLINK gen3x12 (600GB/s)



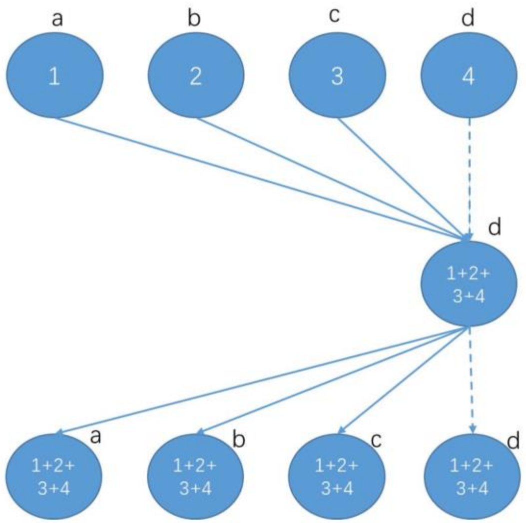
COLLECTIVE COMMUNICATION [3]

Multiple senders and/or receivers

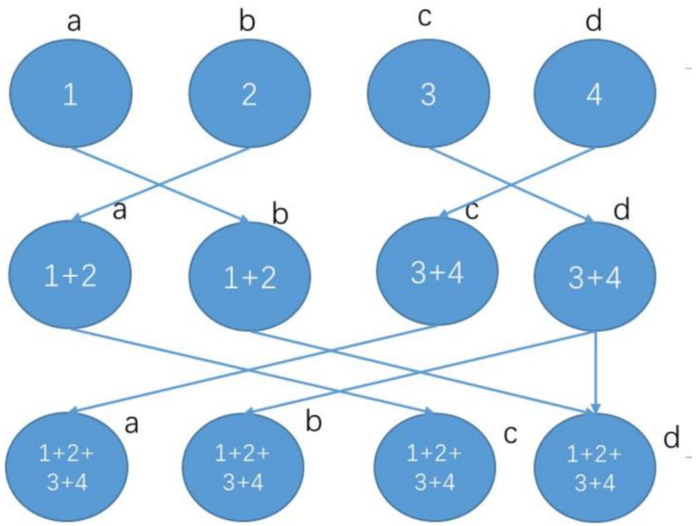


How to do all reduce ?

- All-reduce algorithms [4]



Reduce and broadcast



Butterfly reduce

Ring all-reduce

NCCL

- Handle intra-node GPU communication in an optimal way
- Supported collectives: broadcast, all-gather, reduce, all-reduce, reduce-scatter
- Key features: single-node, up to 8 GPUs, non-blocking interface, multi-process support,
- Implementation: monolithic CUDA C++ kernel

Gloo

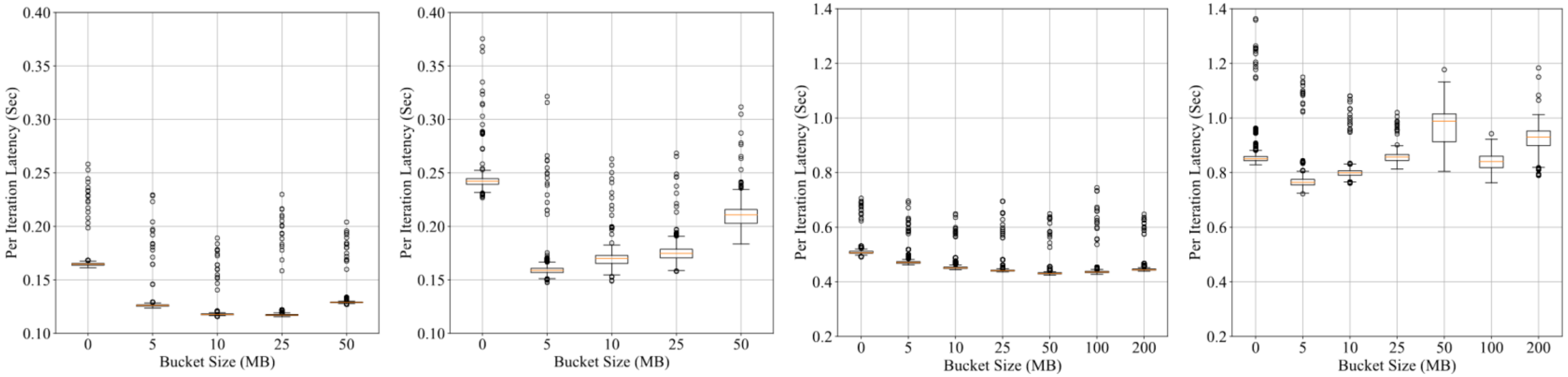
- Gloo is a collective communications library [6]
- It comes with a number of collective algorithms useful for machine learning applications, including a barrier, broadcast, and allreduce.
- Optimized for CPU, support Float16
- Facebook paper “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour” [7]
  - with gloo and Caffe2
  - 256 GPUs (32 servers)
  - Minibatch size 8192 no loss of accuracy

Communication backends list [5]

Backend	Comm. Functions	Optimized for	Float32	Float16
MPI	All	CPU, GPU	Yes	No
GLOO	All (on CPU), broadcast & all-reduce (on GPU)	CPU	Yes	Yes
NCCL	broadcast, all reduce, reduce and all gather (on GPU)	GPU only	Yes	Yes

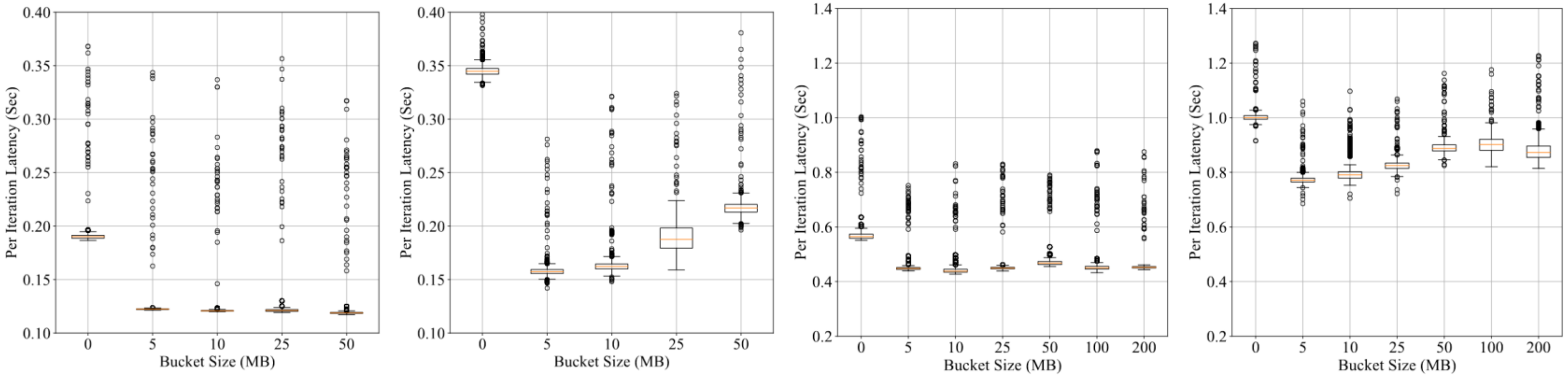
Performance Comparison

"PyTorch Distributed: Experiences on Accelerating Data Parallel Training"



(a) ResNet50 on NCCL (b) ResNet50 on Gloo (c) BERT on NCCL (d) BERT on Gloo

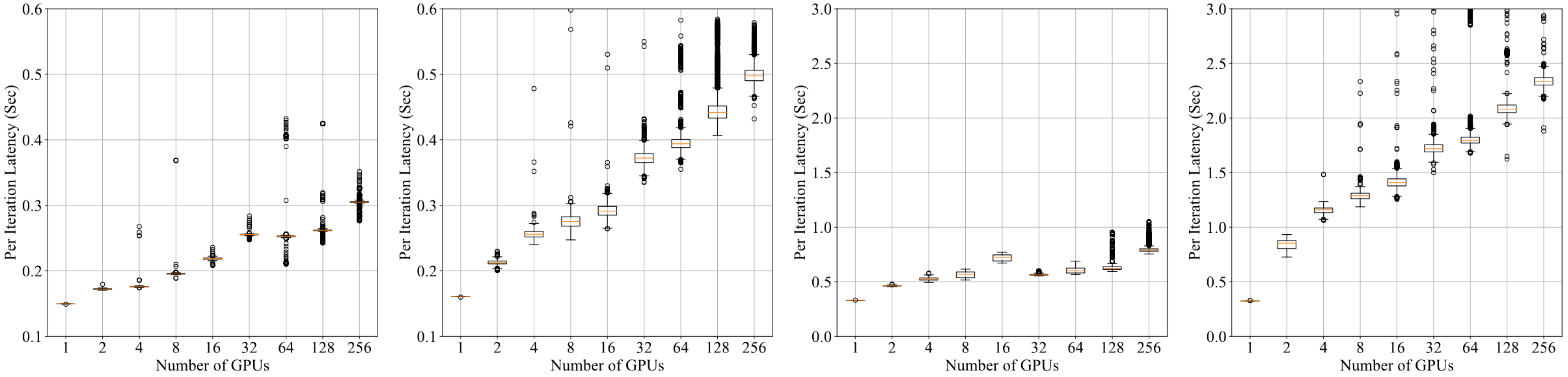
Figure 7: Per Iteration Latency vs Bucket Size on 16 GPUs



(a) ResNet50 on NCCL (b) ResNet50 on Gloo (c) BERT on NCCL (d) BERT on Gloo

Figure 8: Per Iteration Latency vs Bucket Size on 32 GPUs

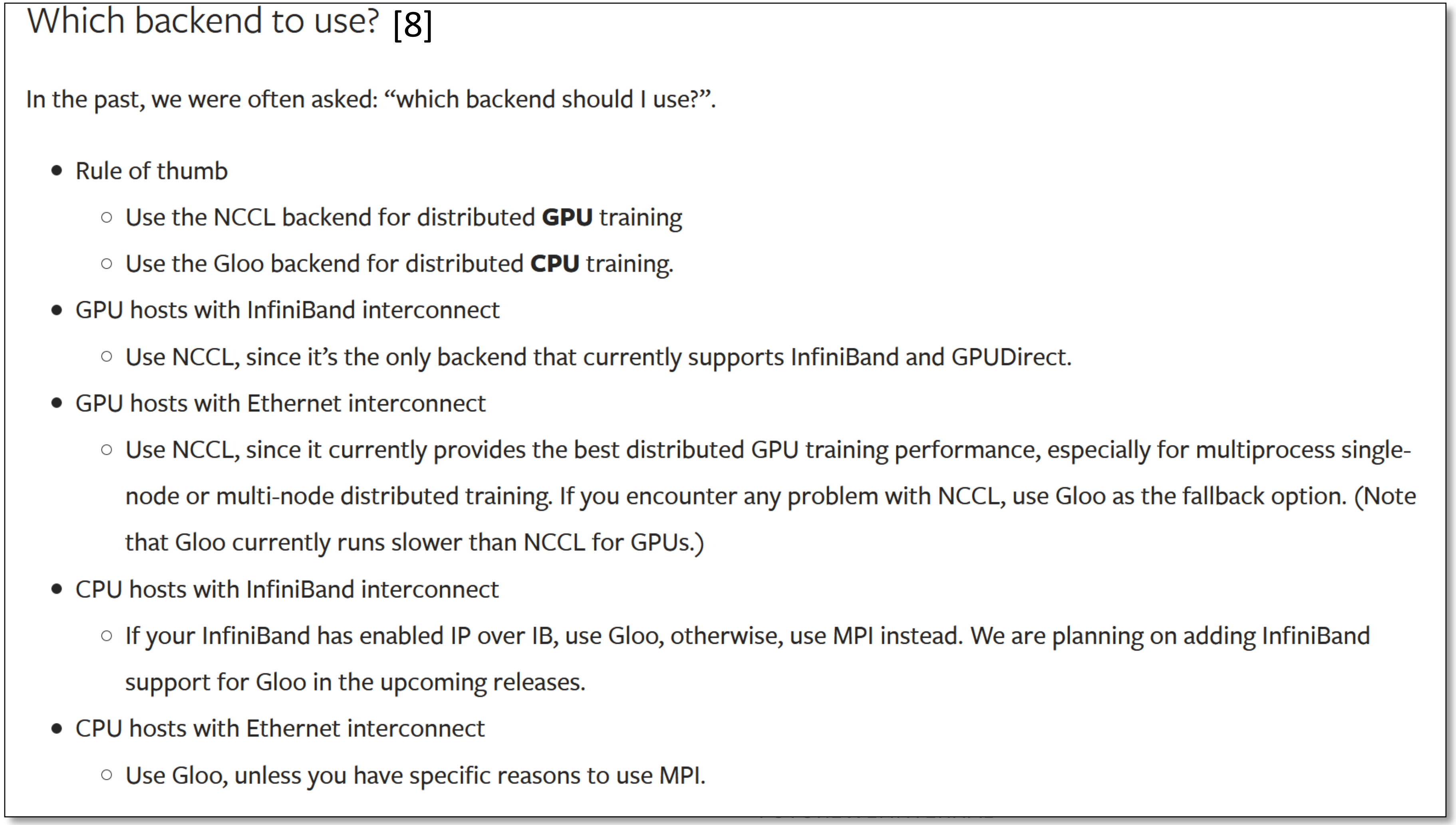
Scalability Comparison



(a) ResNet50 on NCCL (b) ResNet50 on Gloo (c) BERT on NCCL (d) BERT on Gloo

Figure 9: Scalability

Pytorch Suggestions



Which backend to use? [8]

In the past, we were often asked: "which backend should I use?".

- Rule of thumb
  - Use the NCCL backend for distributed **GPU** training
  - Use the Gloo backend for distributed **CPU** training.
- GPU hosts with InfiniBand interconnect
  - Use NCCL, since it's the only backend that currently supports InfiniBand and GPUDirect.
- GPU hosts with Ethernet interconnect
  - Use NCCL, since it currently provides the best distributed GPU training performance, especially for multiprocess single-node or multi-node distributed training. If you encounter any problem with NCCL, use Gloo as the fallback option. (Note that Gloo currently runs slower than NCCL for GPUs.)
- CPU hosts with InfiniBand interconnect
  - If your InfiniBand has enabled IP over IB, use Gloo, otherwise, use MPI instead. We are planning on adding InfiniBand support for Gloo in the upcoming releases.
- CPU hosts with Ethernet interconnect
  - Use Gloo, unless you have specific reasons to use MPI.

References

1. Characterizing Deep Learning Training Workloads on Alibaba-PAI
2. PyTorch Distributed: Experiences on Accelerating Data Parallel Training
3. NCCL, https://images.nvidia.com/events/sc15/pdfs/NCCL-Woolley.pdf
4. All reduce algorithms, https://zhuanlan.zhihu.com/p/79030485
5. MLBench, https://mlbench.github.io/2020/09/08/communication-backend-comparison/
6. Gloo, https://github.com/facebookincubator/gloo
7. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour
8. Pytorch, https://pytorch.org/docs/stable/distributed.html