Multi-Node Training

Bottlenecks of Single Node Jobs

- limited number of GPUs per node
- limited PCI bus bandwidth
- limited CPU power
- limited local storage

Different Kinds of "Multi-Node Jobs"

Model Size:

- model fits into single GPU
- model needs multiple GPUs but fits on single node
- model needs multiple nodes

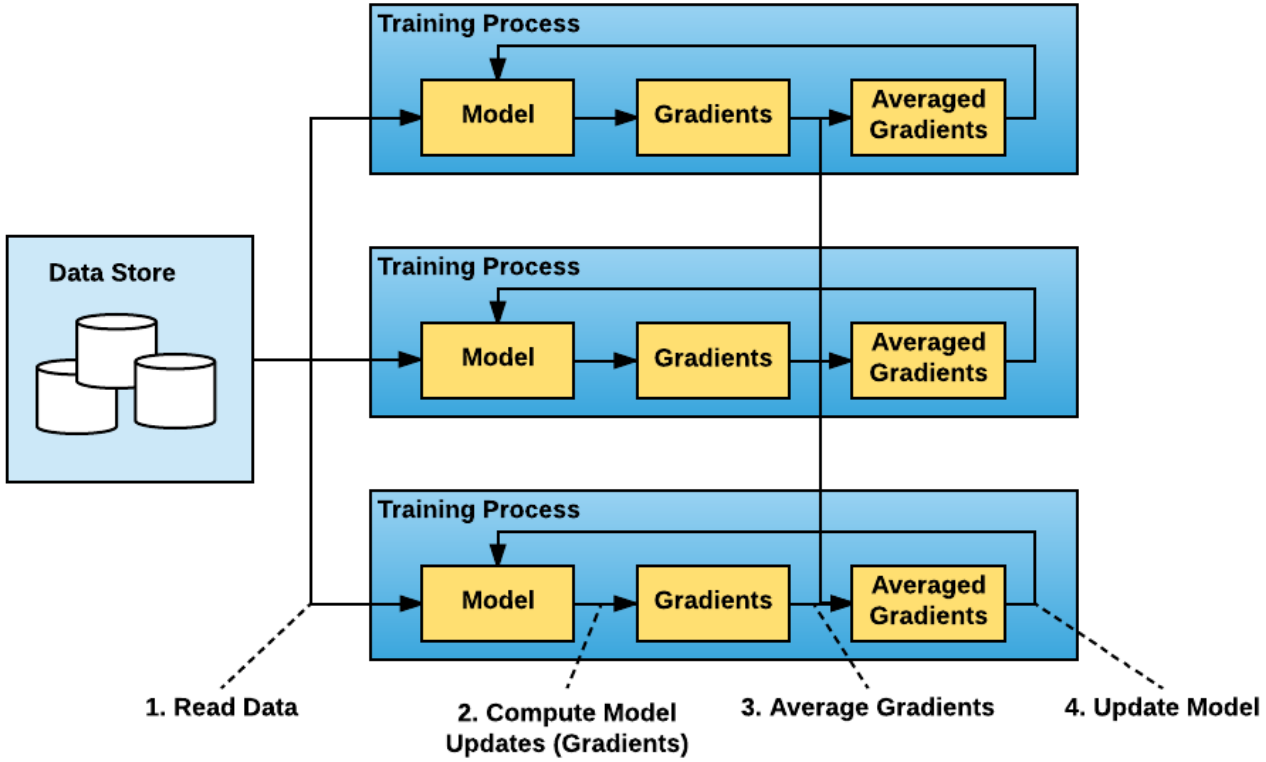
Model Replication:

- model is replicated across multiple GPUs on a single node
- model is replicated across multiple nodes

Multiple Communications Channels

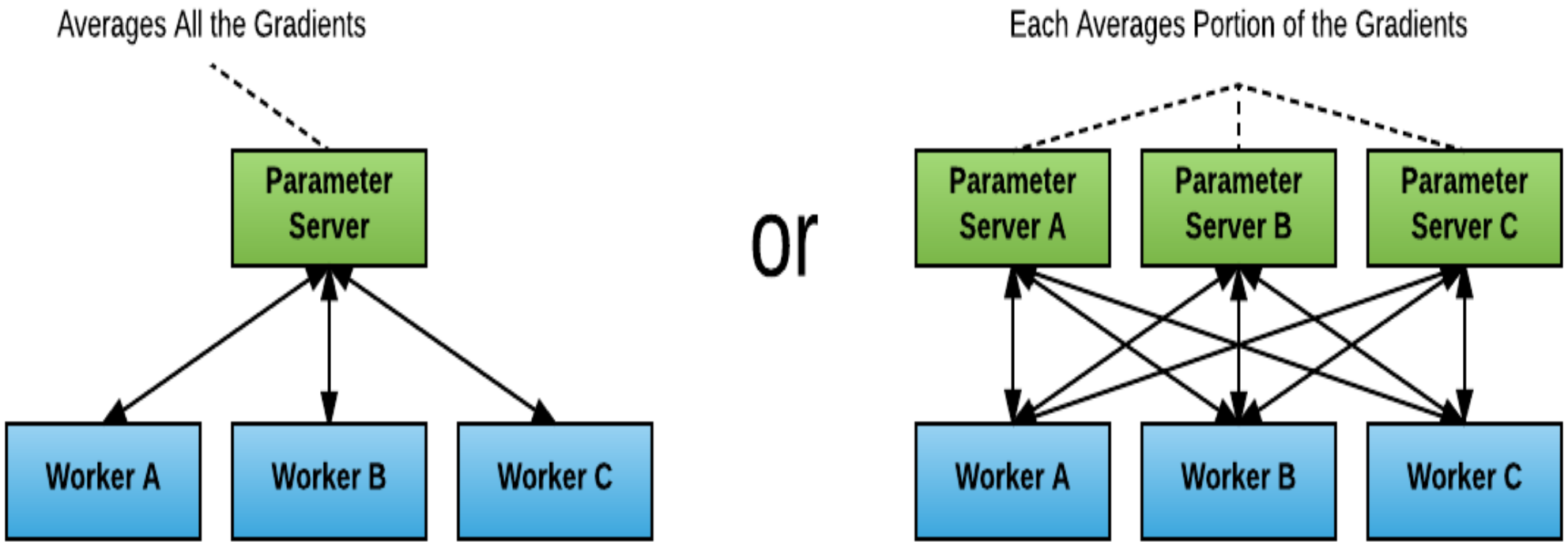
- PCI bus for CPU, CPU↔GPU (computations involving single model)
- RDMA to CPU memory (data)
- GPUdirect for I/O to GPU, RDMA (data loading, inter-node parameters)
- NVLINK between GPUs (multi-GPU SGD within a node)
- standard Ethernet to CPU memory (control messages, maybe data)

Common Multi-GPU / Multi-Node Training



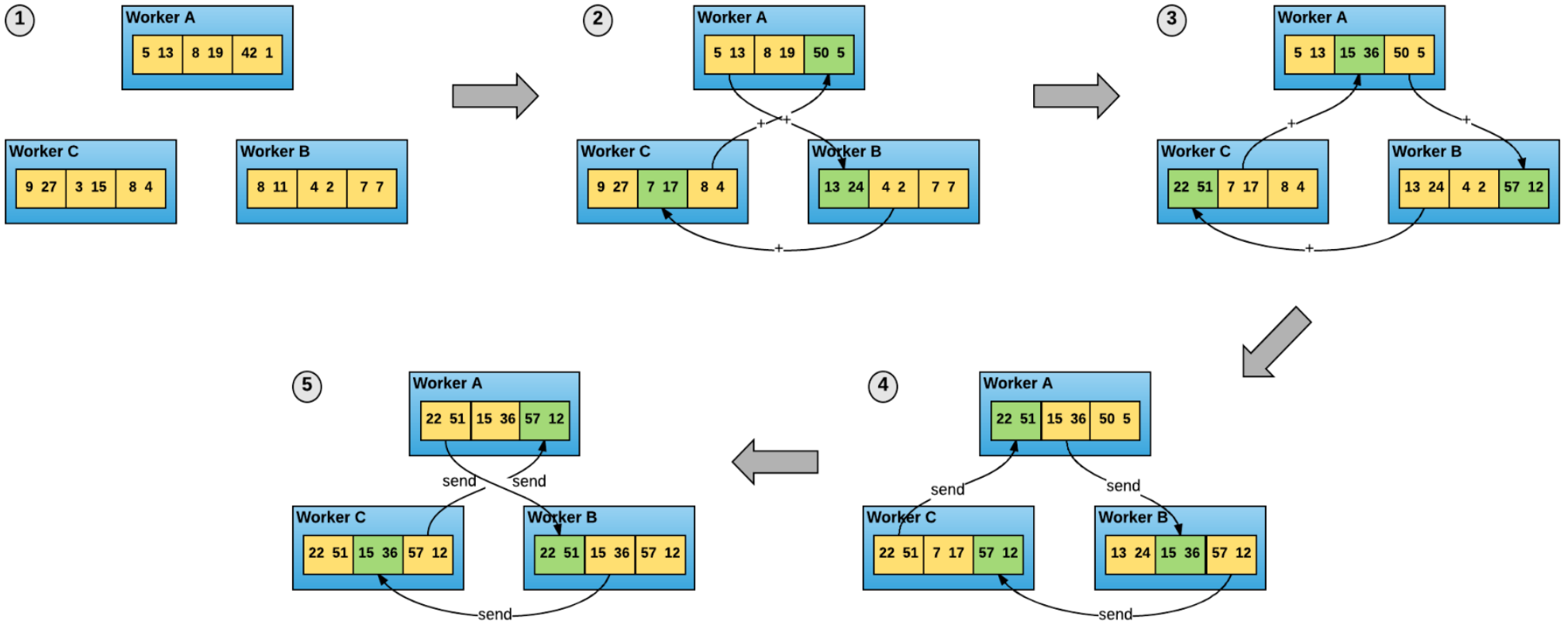
(source: Horovod)

Parameter Servers



(source: Horovod)

Direct Parameter Exchanges



(source: Horovod)

PyTorch Support for Multi-GPU Training

- `DataParallel`, multiple I/O streams
- `DistributedDataParallel`, single node, multiple I/O streams
- `DistributedDataParallel`, multiple nodes, multiple I/O streams

These are wrappers around models and will take care of synchronizing gradients/weights across model instances.

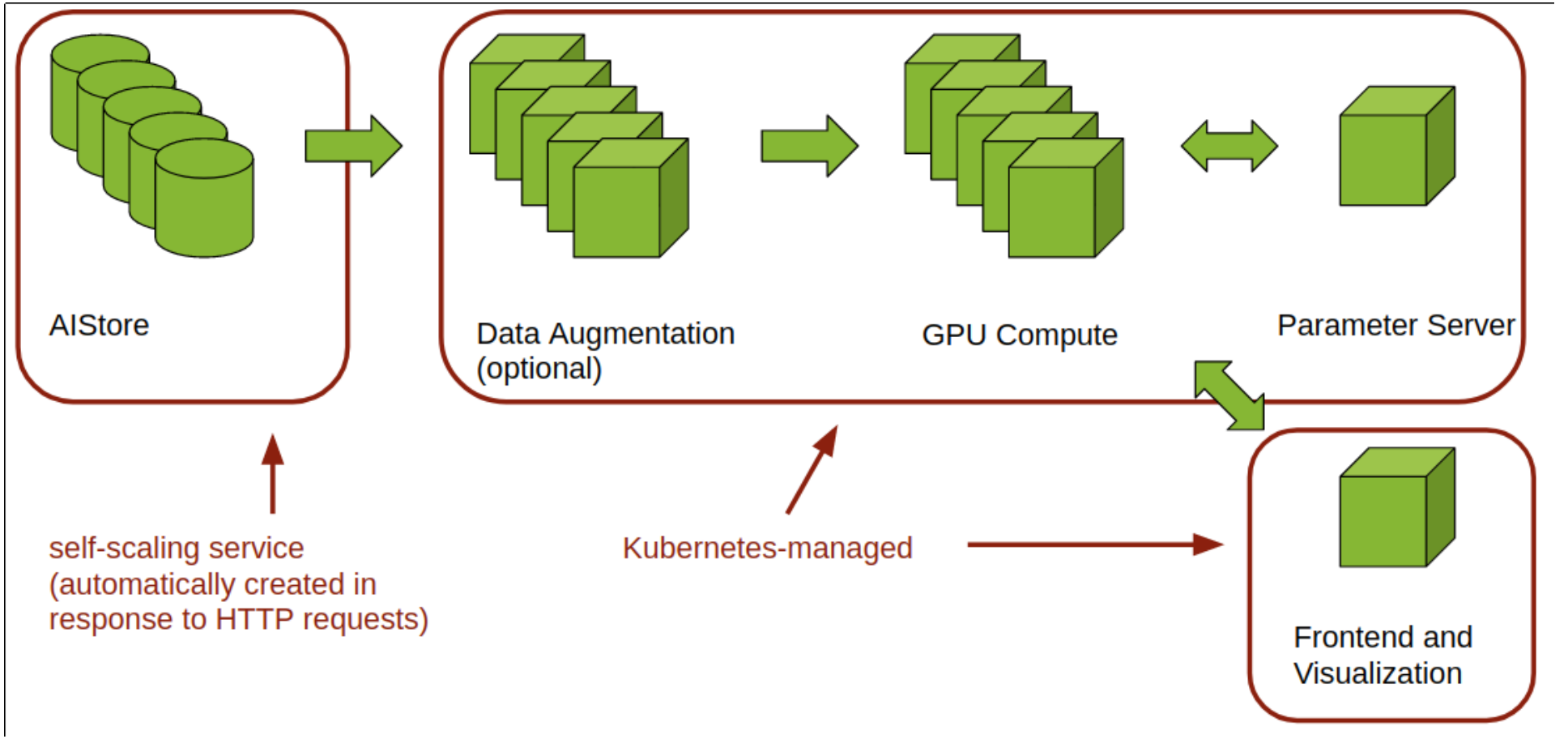
How do you distribute?

- starting up jobs across clusters / machines
  - Ansible
  - Kubernetes
  - Slurm
- communications libraries
  - sockets, ZMQ
  - NCCL, Gloo
  - torch.distributed, Horovod
- all-in-one
  - MPI

Kubernetes

- start and stop processes on a cluster of machines
- encapsulate processes in containers
- provide a virtual network overlay
- provide access to hardware resources
- provide job and pod management

Kubernetes for DL



Kubernetes Pod

- Kubernetes jobs/daemons/services are specified in YAML.
- They are applied with the `kubectl apply` command
- Basic unit is a Pod, a collection of Docker containers ("a pod of whales")

```
apiVersion: v1
kind: Pod
metadata:
name: mypod
spec:
containers:
- name: mypod
    image: gcr.io/research-191823/bigdata19
    command: ["nvidia-smi"]
    resources:
    limits:
        nvidia.com/gpu: "1"
restartPolicy: Never
```

Kubernetes Setup with `kubetpl`

Job specs are complex and contain many repeats; templating helps:

GPU-based Job:

```
$ kubetpl job -G 1 -c nvidia-smi | kubectl apply -f -
```

CPU-based Pod:

```
$ kubetpl pod -l tmbdev/redis-server -c redis-server | kubectl apply -f -
```

Distributed Training

(notebook)