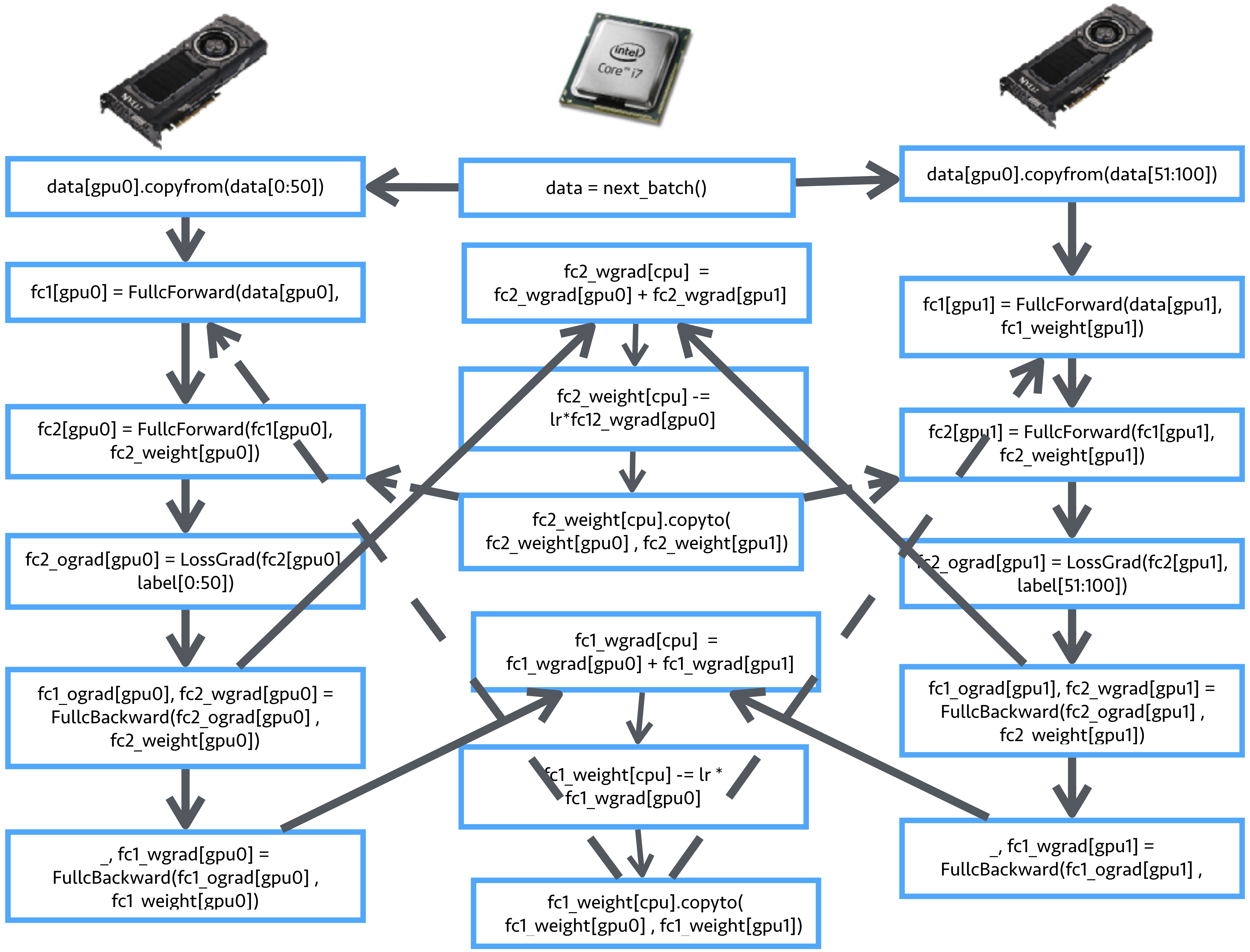
Writing Parallel Programs is Painful

2-layer neural networks with 2 GPUs



data = next_batch()

data[gpu0].copyfrom(data[0:50])

fc1[gpu0] = FullcForward(data[gpu0],

fc2[gpu0] = FullcForward(fc1[gpu0], fc2_weight[gpu0])

fc2_ograd[gpu0] = LossGrad(fc2[gpu0] label[0:50])

fc1_ograd[gpu0], fc2_wgrad[gpu0] = FullcBackward(fc2_ograd[gpu0] , fc2_weight[gpu0])

_, fc1_wgrad[gpu0] = FullcBackward(fc1_ograd[gpu0] , fc1_weight[gpu0])

fc2_wgrad[cpu] = fc2_wgrad[gpu0] + fc2_wgrad[gpu1]

fc2_weight[cpu] -= lr*fc12_wgrad[gpu0]

fc2_weight[cpu].copyto(fc2_weight[gpu0] , fc2_weight[gpu1])

fc1_wgrad[cpu] = fc1_wgrad[gpu0] + fc1_wgrad[gpu1]

fc1_weight[cpu] -= lr * fc1_wgrad[gpu0]

fc1_weight[cpu].copyto(fc1_weight[gpu0] , fc1_weight[gpu1])

data[gpu0].copyfrom(data[51:100])

fc1[gpu1] = FullcForward(data[gpu1], fc1_weight[gpu1])

fc2[gpu1] = FullcForward(fc1[gpu1], fc2_weight[gpu1])

fc2_ograd[gpu1] = LossGrad(fc2[gpu1], label[51:100])

fc1_ograd[gpu1], fc2_wgrad[gpu1] = FullcBackward(fc2_ograd[gpu1] , fc2_weight[gpu1])

_, fc1_wgrad[gpu1] = FullcBackward(fc1_ograd[gpu1] ,

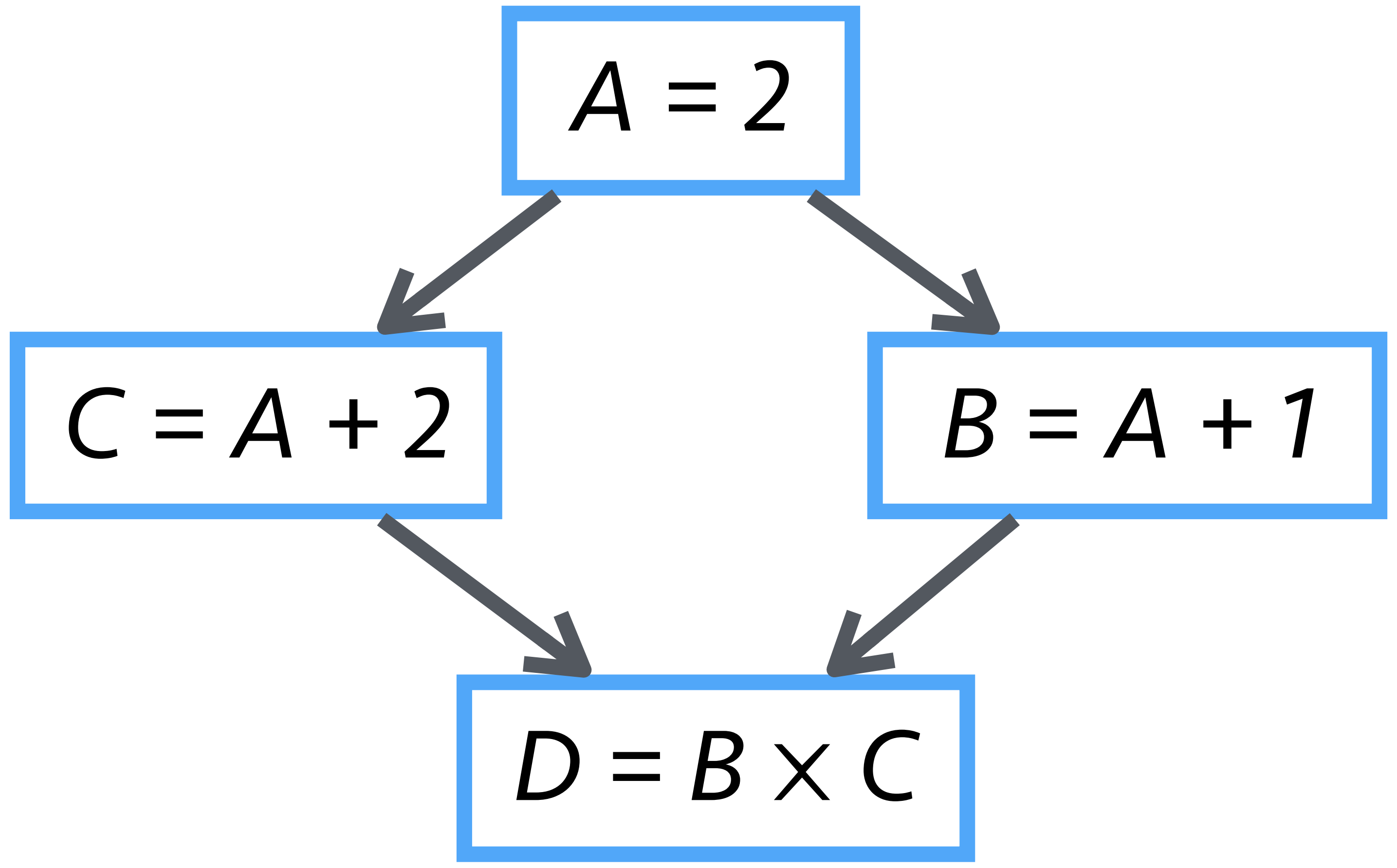
A network may have hundreds of layers

Auto Parallelization

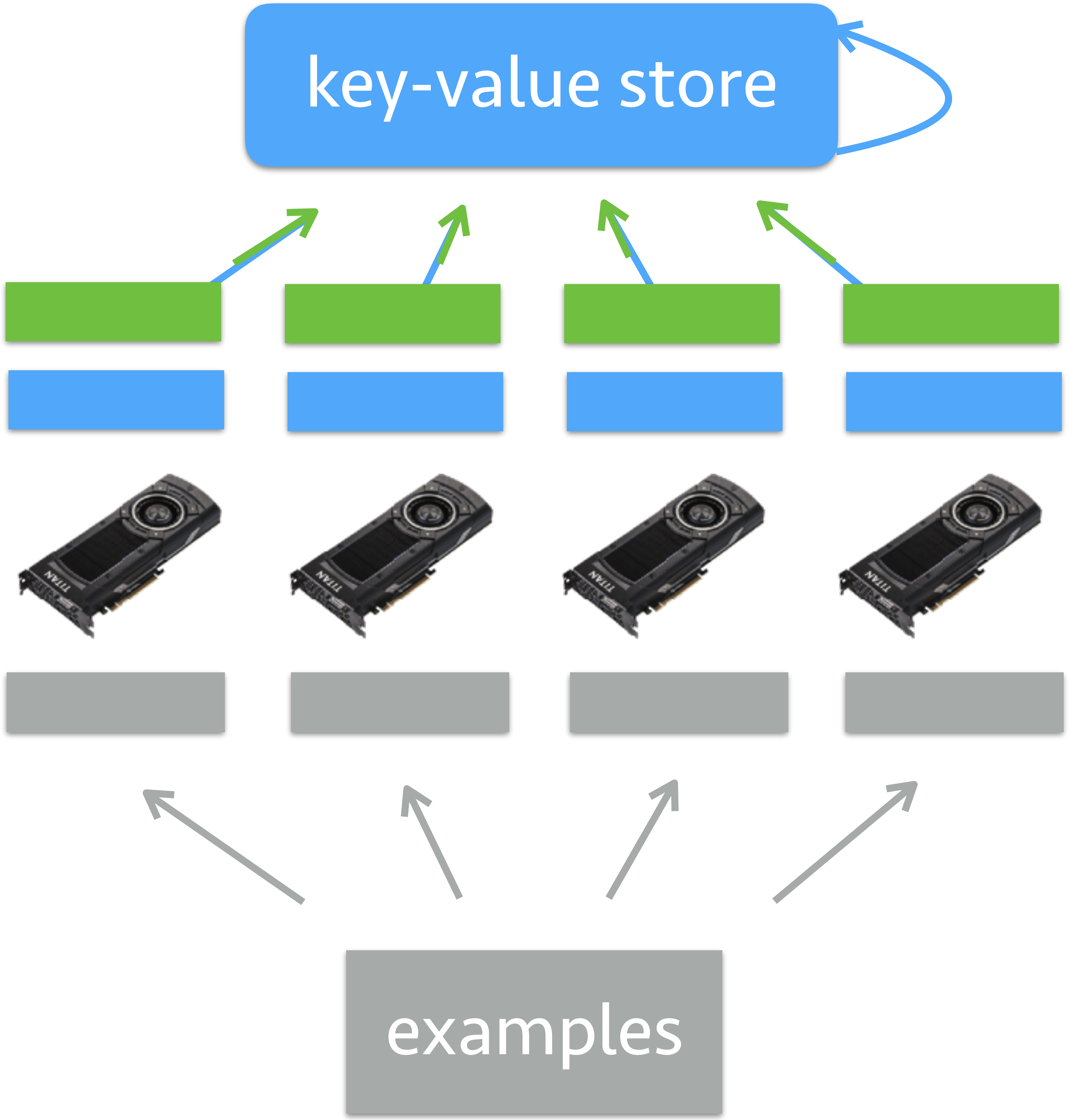
Write **serial** programs

```
>>> import mxnet as mx
>>> A = mx.nd.ones((2,2)) *2
>>> C = A + 2
>>> B = A + 1
>>> D = B * C
```

Run in **parallel**



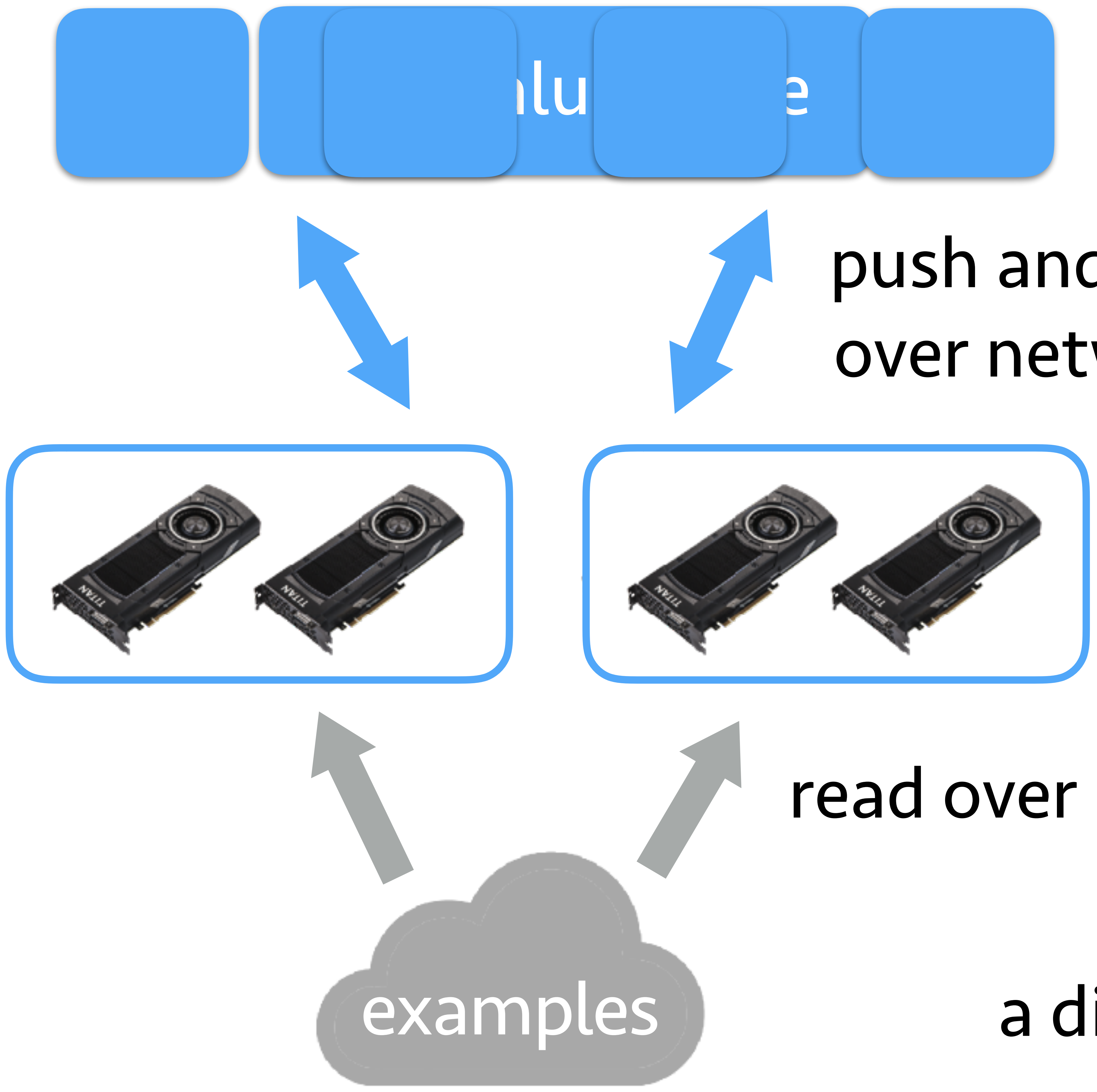
Data Parallelism



1. Read a data partition
2. Pull the parameters
3. Compute the gradient
4. Push the gradient
5. Update the parameters

Distributed Computing

A user does not need to change the codes when using multiple machines



multiple
server machines

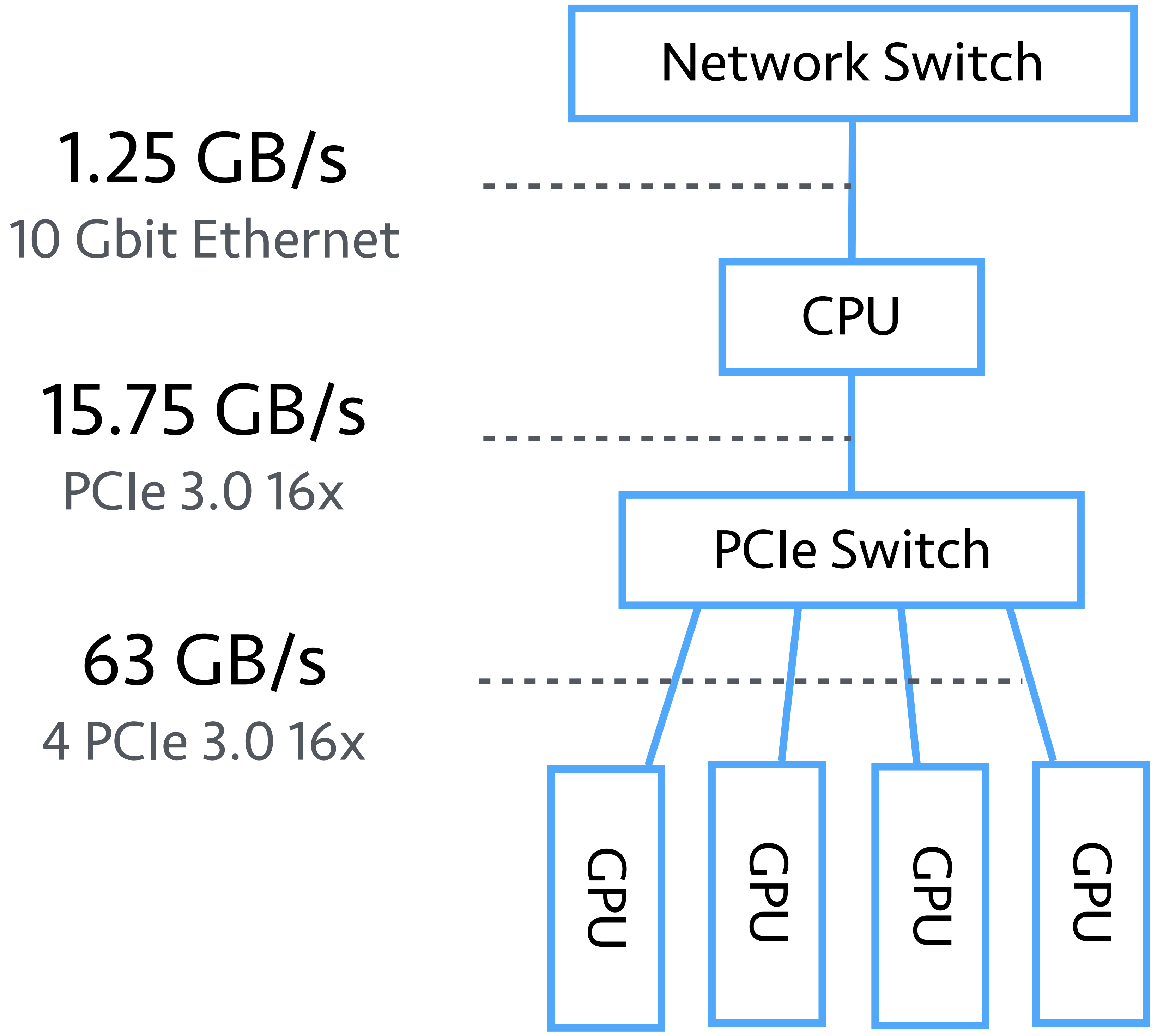
push and pull
over network

multiple
worker machines

read over network

Store data in
a distributed filesystem

Scale to Multiple GPU Machines



Network Switch

1.25 GB/s
10 Gbit Ethernet

CPU

15.75 GB/s
PCIe 3.0 16x

PCIe Switch

63 GB/s
4 PCIe 3.0 16x

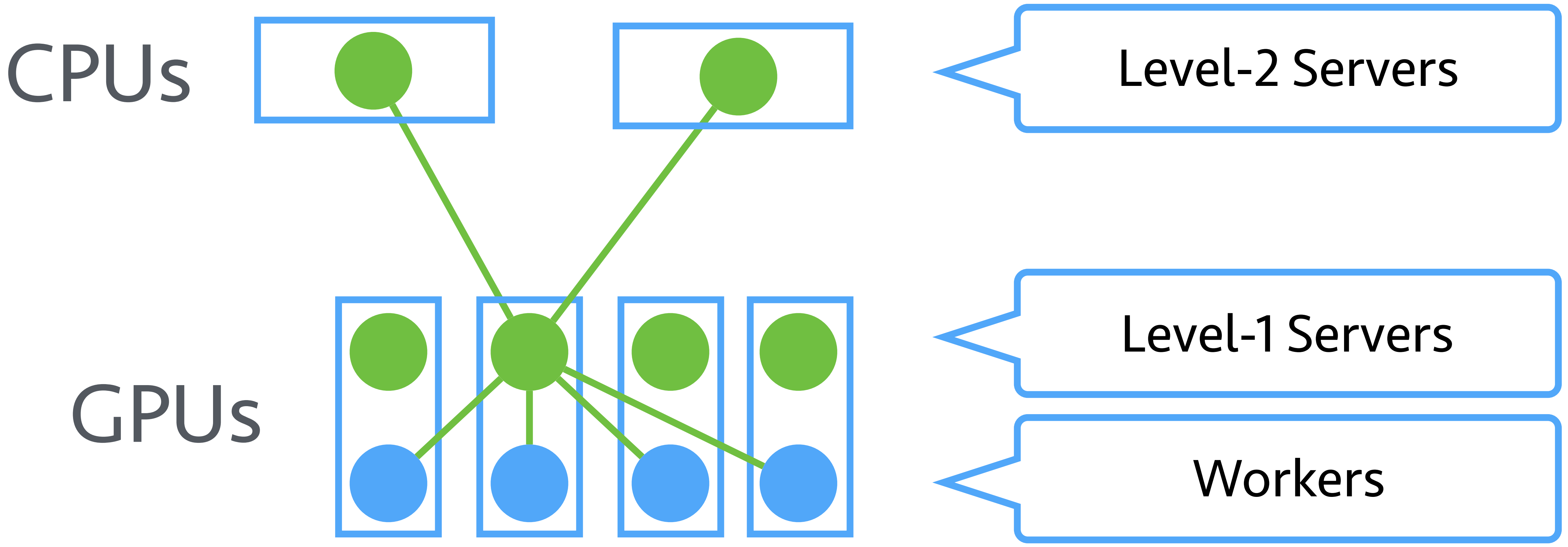
GPU

GPU

GPU

GPU

Hierarchical parameter server



CPUs

GPUs

Level-2 Servers

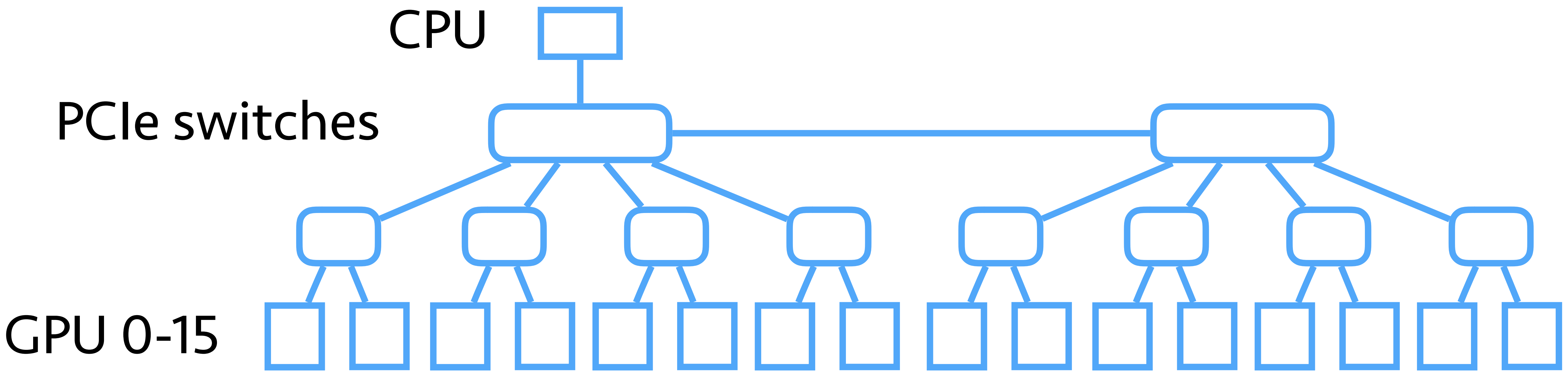
Level-1 Servers

Workers

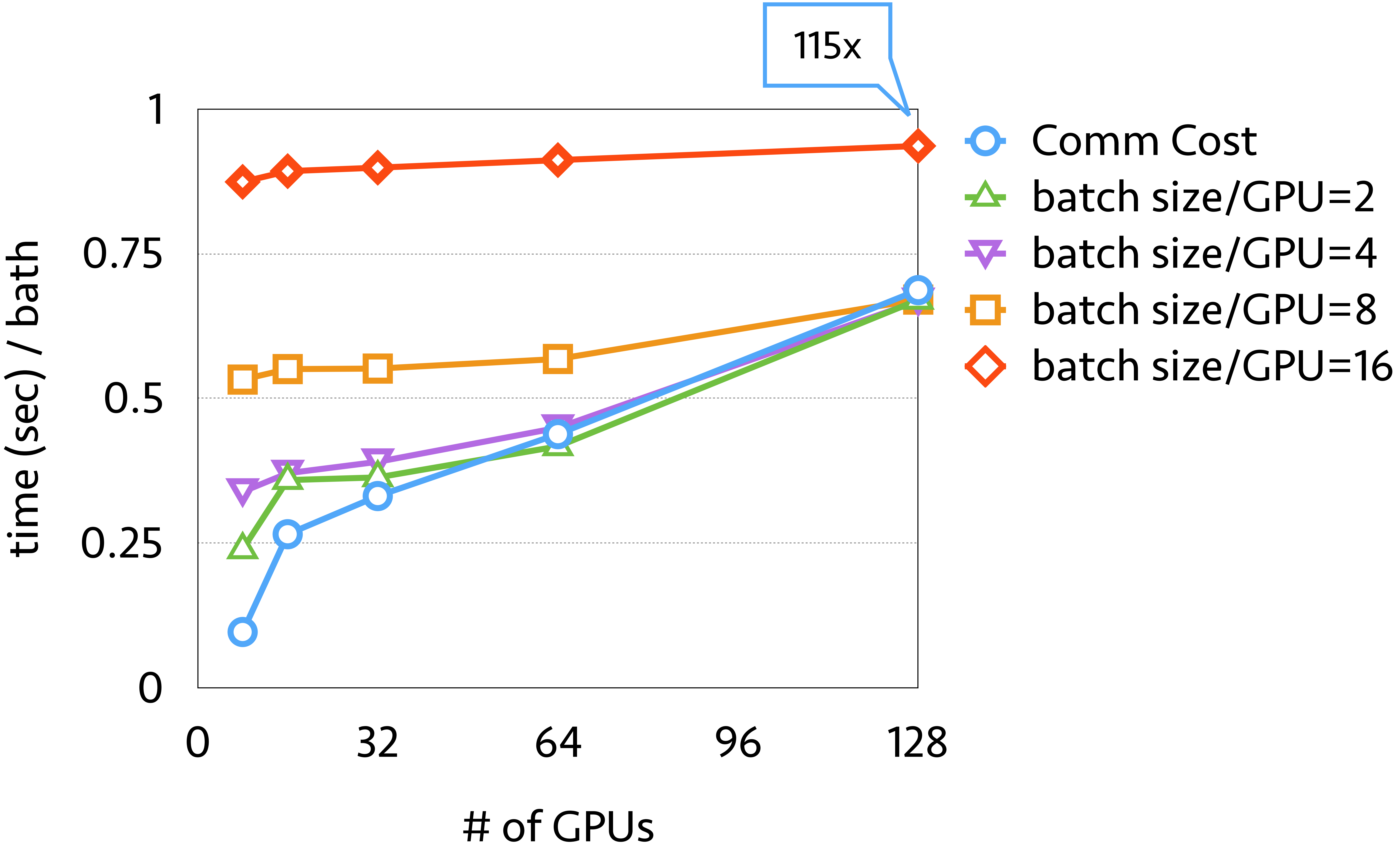
Experiment Setup

- IMAGENET
  - ✓ 1.2 million images with 1000 classes
- Resnet 152-layer model
- EC2 P2.16xlarge

- Minibatch SGD
- Synchronized Updating

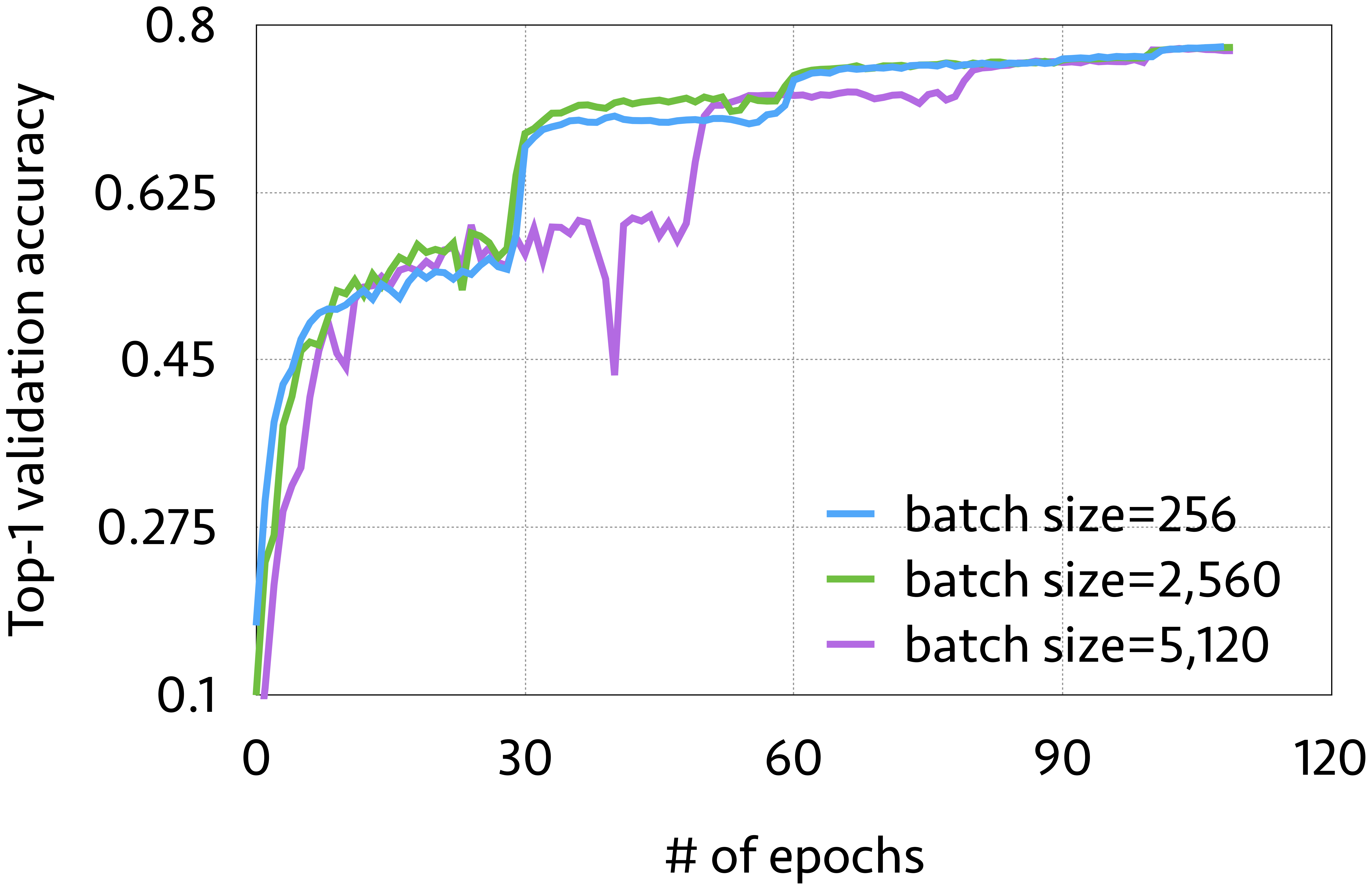


Scalability over Multiple Machines

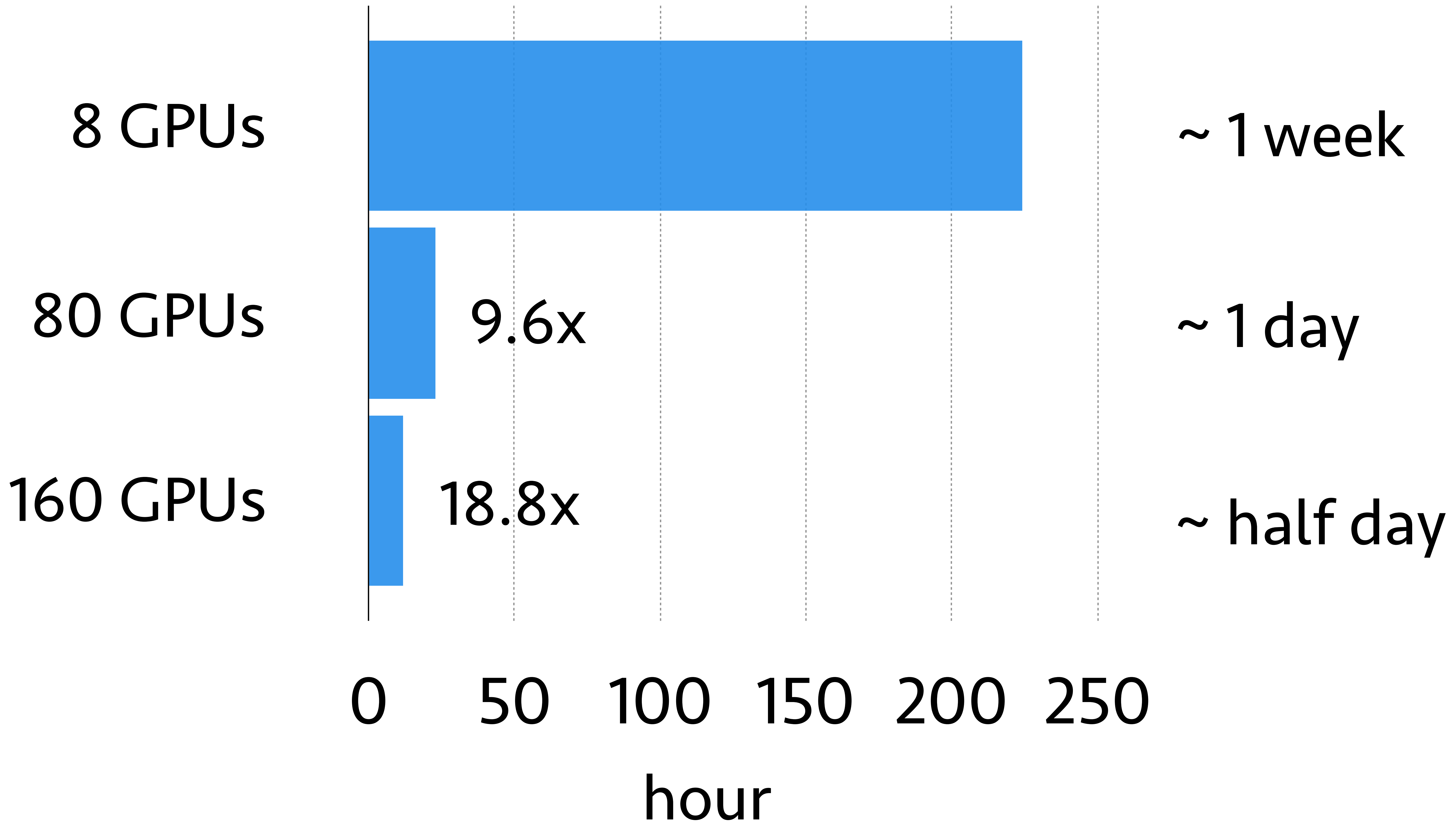


Convergence

- Increase learning rate by 5x
- Increase learning rate by 10x, decrease it at epoch 50, 80



Time to achieve 22.5% top-1 accuracy


8 GPUs
80 GPUs
160 GPUs
9.6x
18.8x
~ 1 week
~ 1 day
~ half day
hour