Horovod: Distributed training framework

Khemraj Shukla

Division of Applied Mathematics, Brown University

Outline

- What's Horovod?

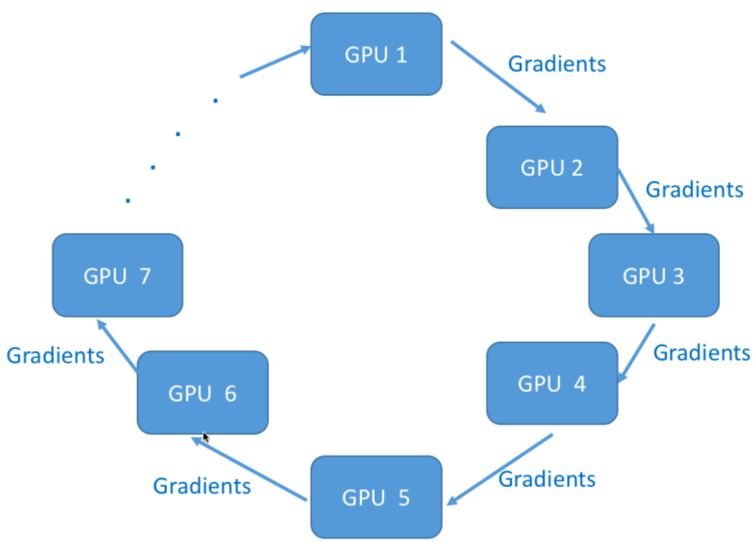
- How to use?
  1. Data parallelism
  2. Ensemble training

- Examples for function approximation/PINN
  1. Data parallelism
  2. Ensemble training

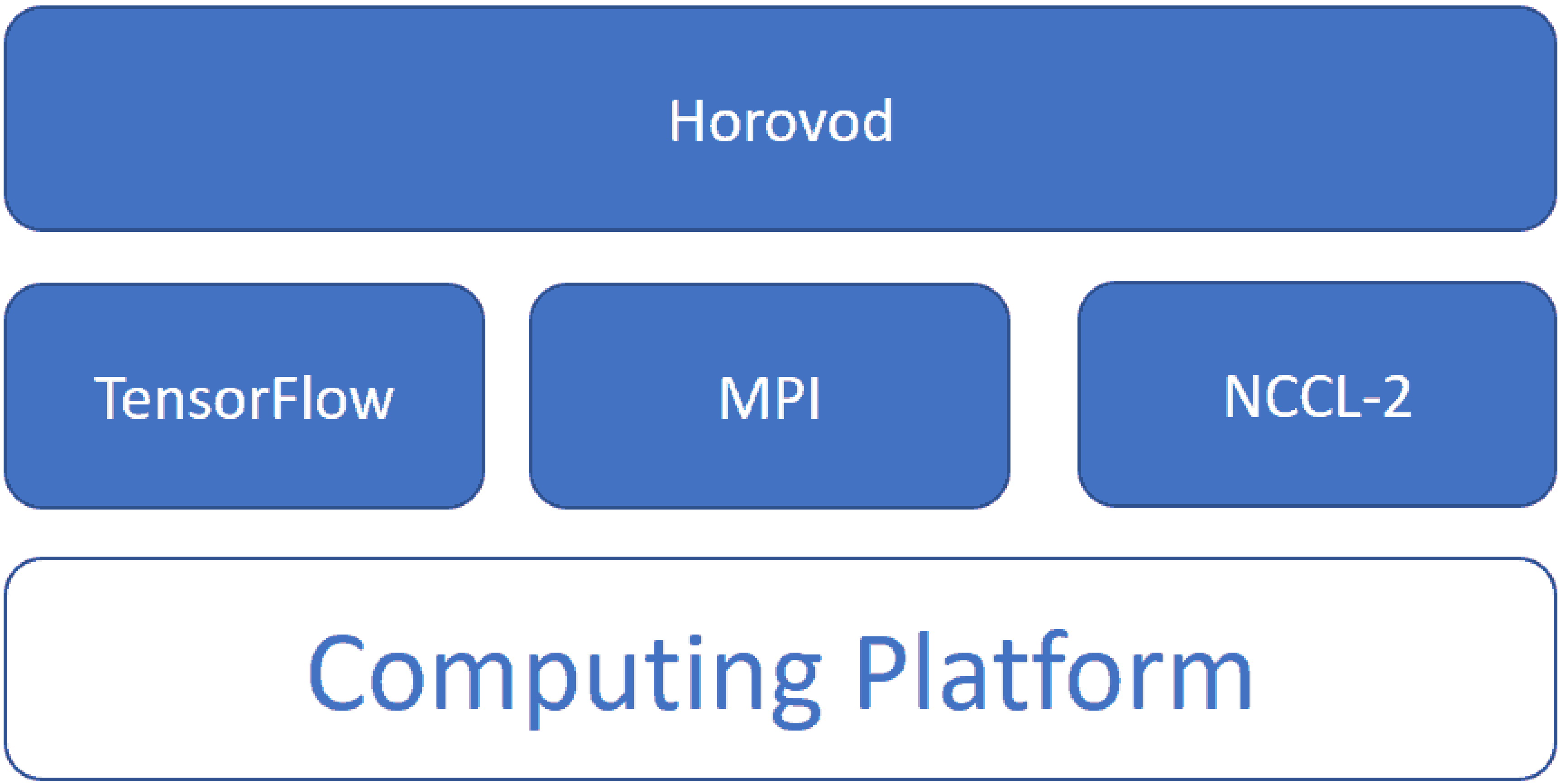
What's Horovod?

What’s Horovod?

What's Horovod?

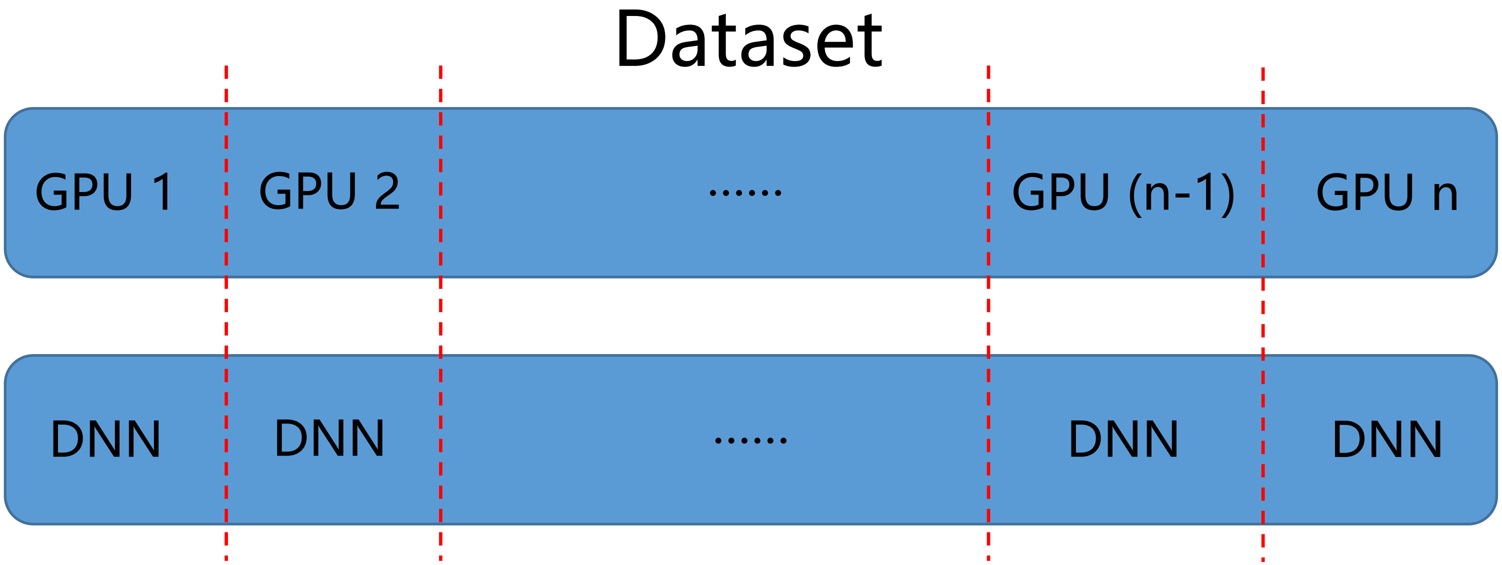


What's Horovod?

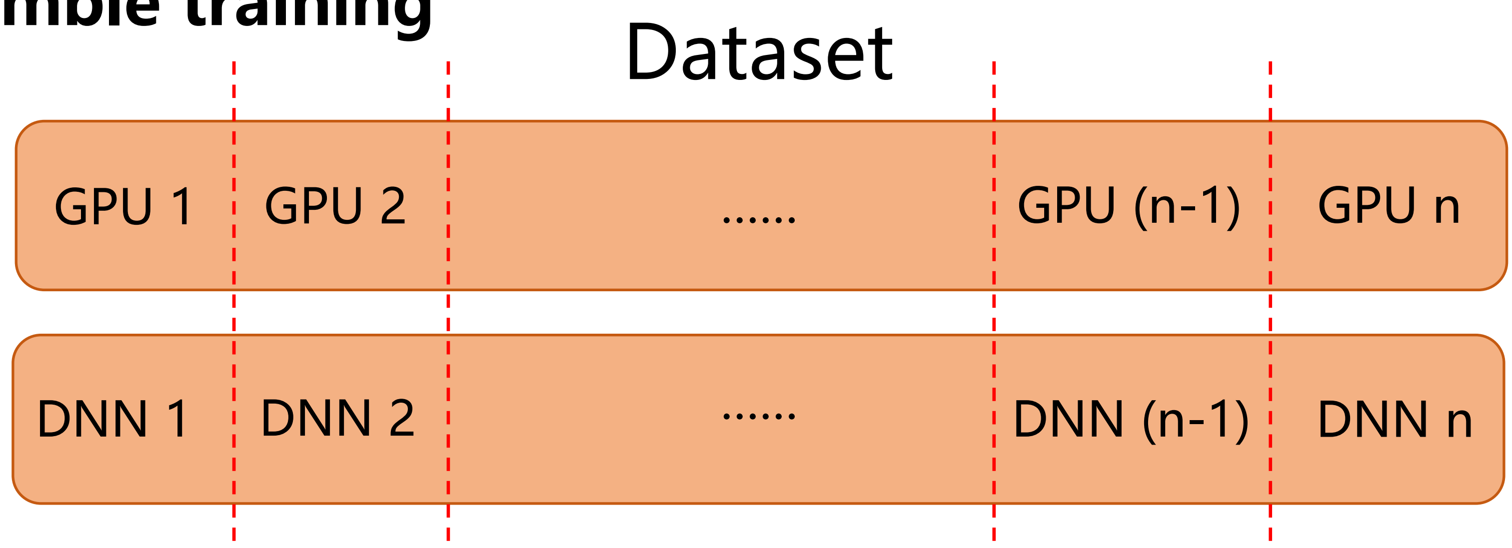


Data parallelism & Ensemble training

Data parallelism



Ensemble training



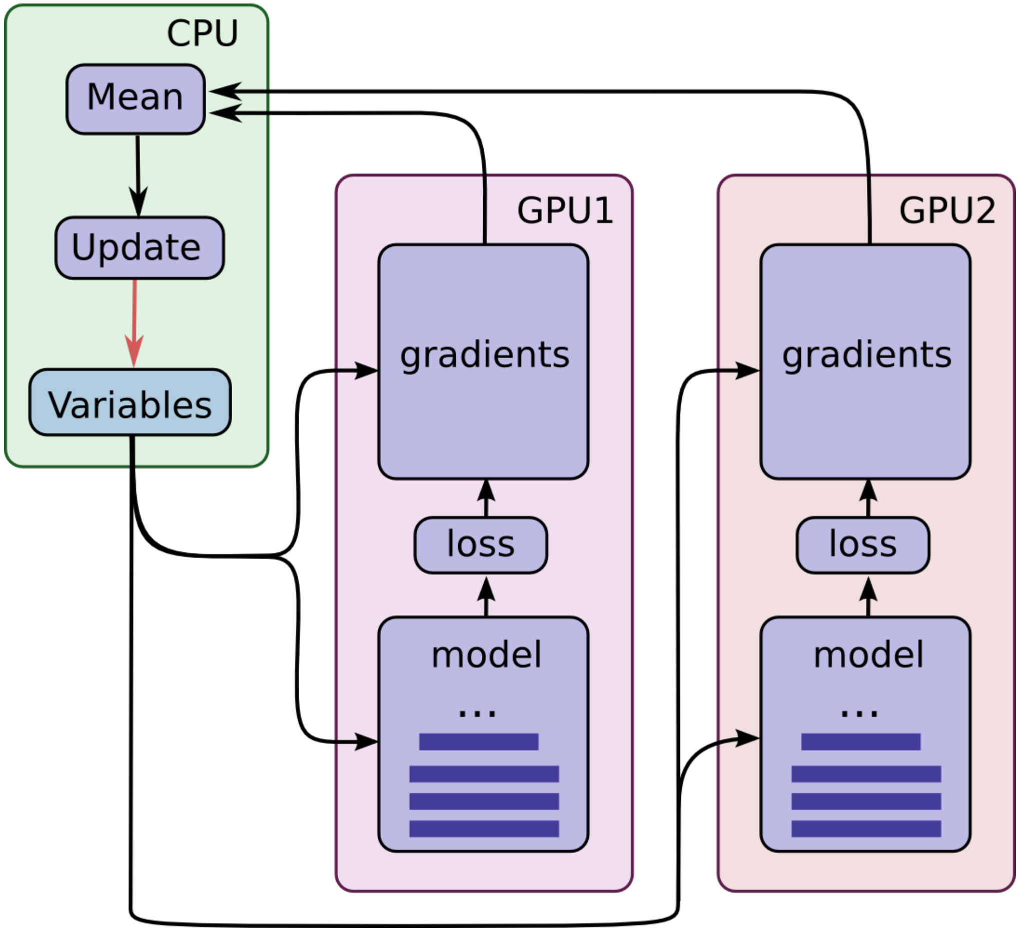
Outline

- What's Horovod?

- How to use?
  1. Data parallelism
  2. Ensemble training

- Examples for function approximation/PINN
  1. Data parallelism
  2. Ensemble training

How to use Horovod – Data parallelism



1. Run multiple copies of the training scripts, and each copy:
   - Reads a chunk of data
   - Train the model
   - Compute the gradients
2. Average the gradients from all the copies
3. Update the model
4. Repeat 1-3

Show me the code: Data parallelism

```
import tensorflow as tf
import horovod.tensorflow ad hvd

hvd.init() #initialize the environment

#pin one GPU to each tensorflow process
config = tf.ConfigProto()
config.gpu_options.visible_device_list
= str(hvd.local_rank())

if hvd.rank() == 0: #read data
        read_chunk_0
else:
        read_chunk_n…

# Build model…
loss = …
opt = tf.train.AdamOptimizer(learning_rate *
hvd.size())

# Add Horovod Distributed Optimizer
opt = hvd.DistributedOptimizer(opt)

train = opt.minimize(loss)

init = tf.global_variables_initializer()
sess.run(init)

#broadcast the initialization to all processes
bcast = hvd.broadcast_global_variables(0)
sess.run(bcast)

while train_step < step_max:
        sess.run([train, loss], feed_dict=…)
```

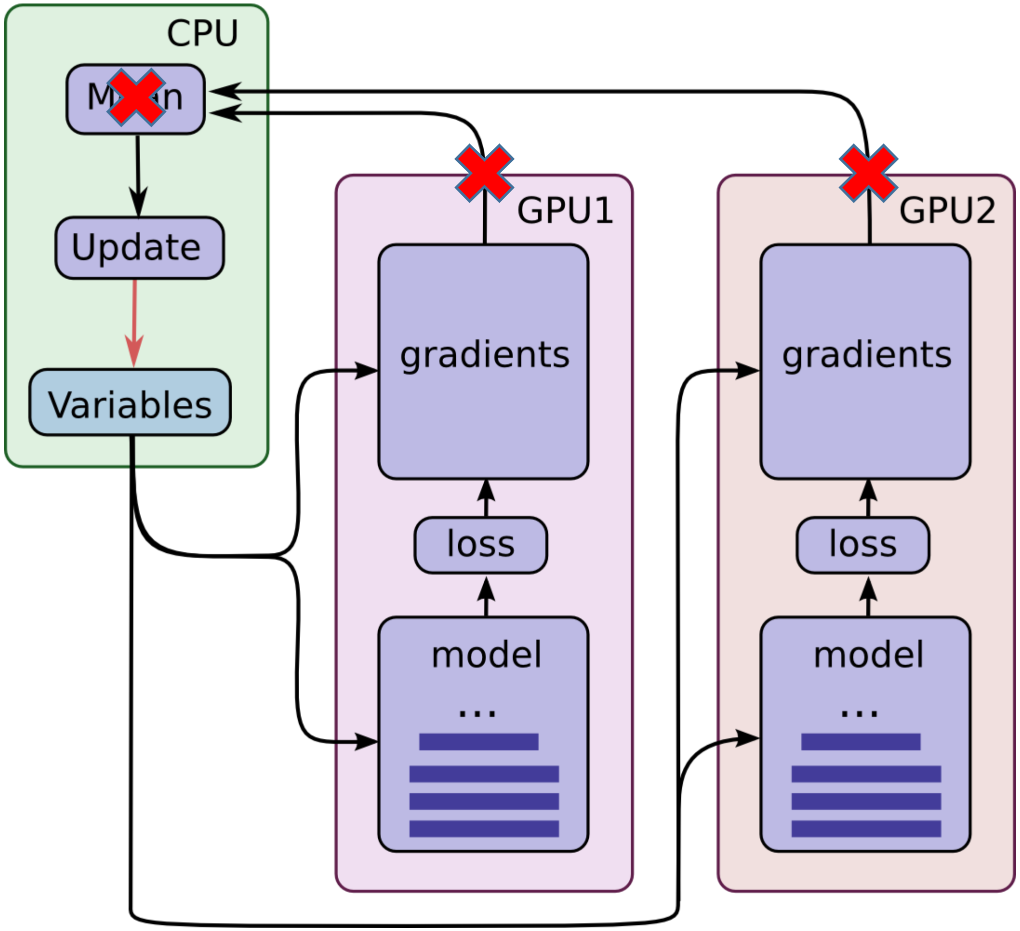
```
horovodrun/mpirun –np NP –H
localhost:np
python *.py

mpirun –np NP –H
server 1:np…server n: np
python *.py
```

Show me the code: Data parallelism

```
import tensorflow as tf
import horovod.tensorflow ad hvd    1

hvd.init()  2

config = tf.ConfigProto()
config.gpu_options.visible_device_list
= str(hvd.local_rank())    3

if hvd.rank() == 0:
        read_chunk_0
else:                    4
        read_chunk_n...

loss = ...
opt = tf.train.AdamOptimizer(learning_rate *
hvd.size())    5

opt = hvd.DistributedOptimizer(opt)    6

train = opt.minimize(loss)
```

```
init = tf.global_variables_initializer()
sess.run(init)

7  bcast = hvd.broadcast_global_variables(0)
   sess.run(bcast)

while train_step < step_max:
        sess.run([train, loss], feed_dict=...)
```

How to use Horovod – Ensemble training



1. Run multiple copies of the training scripts, and each copy:
   - Reads a chunk of data
   - Train the model
   - Compute the gradients
2. Update the model using their own gradient
3. Repeat 1-2

Show me the code: Ensemble training

```
import tensorflow as tf
import horovod.tensorflow ad hvd

hvd.init() #initialize the environment

#pin one GPU to each tensorflow process
config = tf.ConfigProto()
config.gpu_options.visible_device_list
= str(hvd.local_rank())

if hvd.rank() == 0: #read data
        read_chunk_0
else:
        read_chunk_n...

# Build model...
loss = ...
opt = tf.train.AdamOptimizer(learning_rate)

# Add Horovod Distributed Optimizer
#opt = hvd.DistributedOptimizer(opt)

train = opt.minimize(loss)

init = tf.global_variables_initializer()
sess.run(init)

#broadcast the initialization to all processes
bcast = hvd.broadcast_global_variables(0)
sess.run(bcast)

while train_step < step_max:
        sess.run([train, loss], feed_dict=...)
```

Show me the code: Ensemble training

```
import tensorflow as tf
import horovod.tensorflow ad hvd          1

hvd.init()  2


config = tf.ConfigProto()
config.gpu_options.visible_device_list
= str(hvd.local_rank())                   3

if hvd.rank() == 0:
        read_chunk_0
else:                         4
        read_chunk_n...


loss = ...
opt = tf.train.AdamOptimizer(learning_rate)

#opt = hvd.DistributedOptimizer(opt)

train = opt.minimize(loss)
```

```
init = tf.global_variables_initializer()
sess.run(init)


5 bcast = hvd.broadcast_global_variables(0)
  sess.run(bcast)

while train_step < step_max:
        sess.run([train, loss], feed_dict=...)
```

Outline

- What's Horovod?

- How to use?
  1. Data parallelism
  2. Ensemble training

- Examples for function approximation/PINN
  1. Data parallelism
  2. Ensemble training

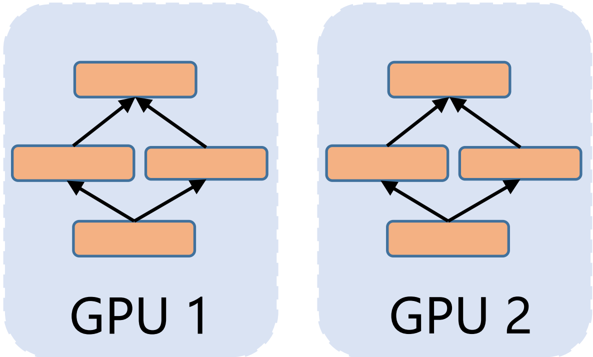
Example 1: Function approximation (Data parallelism)

$$y = \sin(2\pi x) + \sin(4\pi x), x \in [-1, 1]$$

GPU 1 $x = linspace(-1, 0, \mathrm{N})$

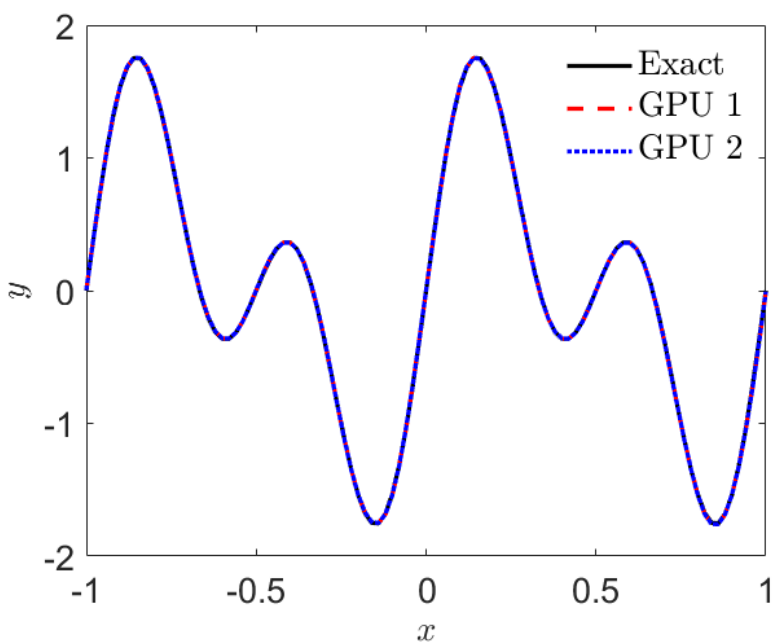
GPU 2 $x = linspace(0, 1, \mathrm{N})$

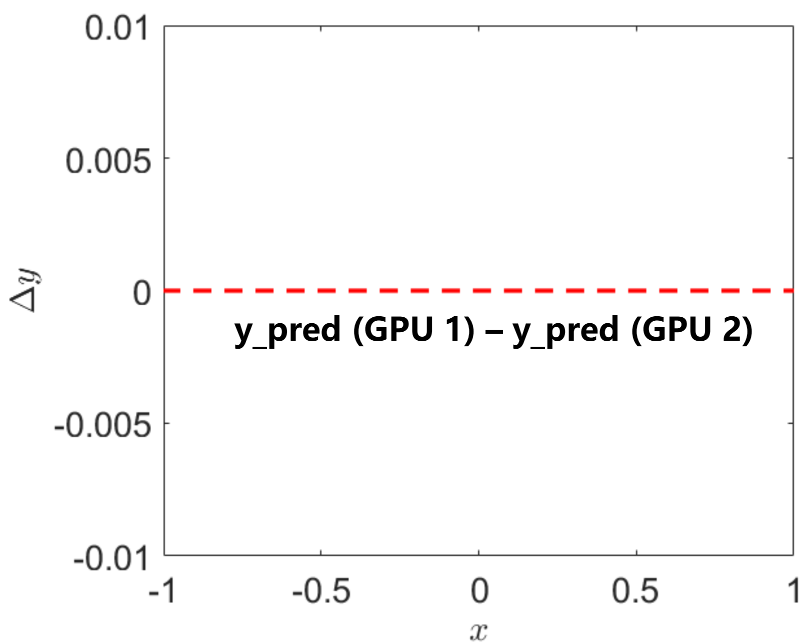
$N = 16$



$depth \times width = 16 \times 2$

$$x_{pred} = linspace(-1, 1, 101)$$





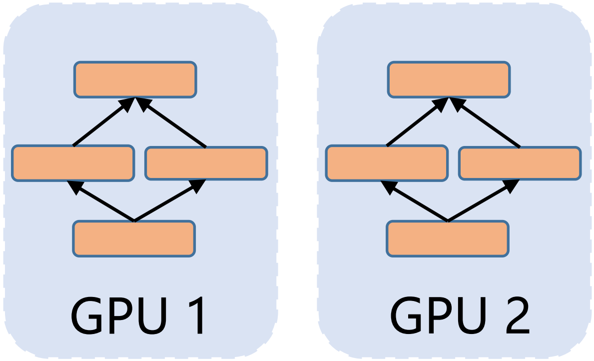
Example 2: Function approximation (Ensemble training)

$$y = \sin(2\pi x) + \sin(4\pi x),\ x \in [-1, 1]$$

GPU 1 $x = linspace(-1, 0, \mathrm{N})$

GPU 2 $x = linspace(0, 1, \mathrm{N})$

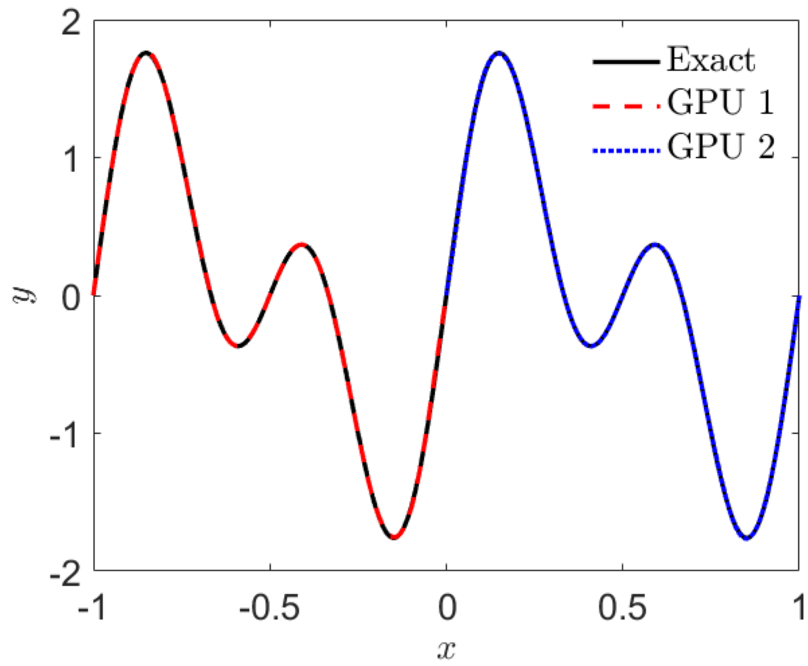
$N = 16$



$depth \times width = 16 \times 2$

GPU 1 $x_{pred} = linspace(-1, 0, 101)$

GPU 2 $x_{pred} = linspace(0, 1, 101)$



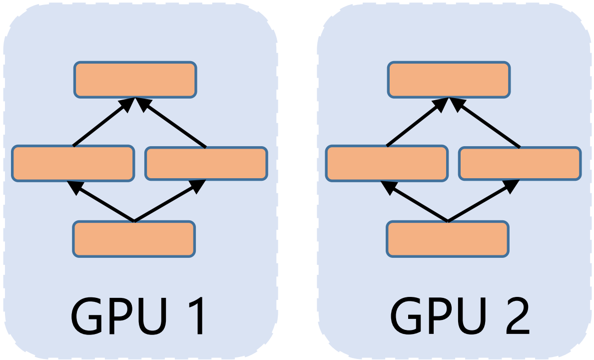
Example 2: Function approximation (Ensemble training)

$$y = \sin(2\pi x) + \sin(4\pi x), x \in [-1,1]$$

GPU 1 $x = linspace(-1, 0, \mathrm{N})$

GPU 2 $x = linspace(0, 1, \mathrm{N})$

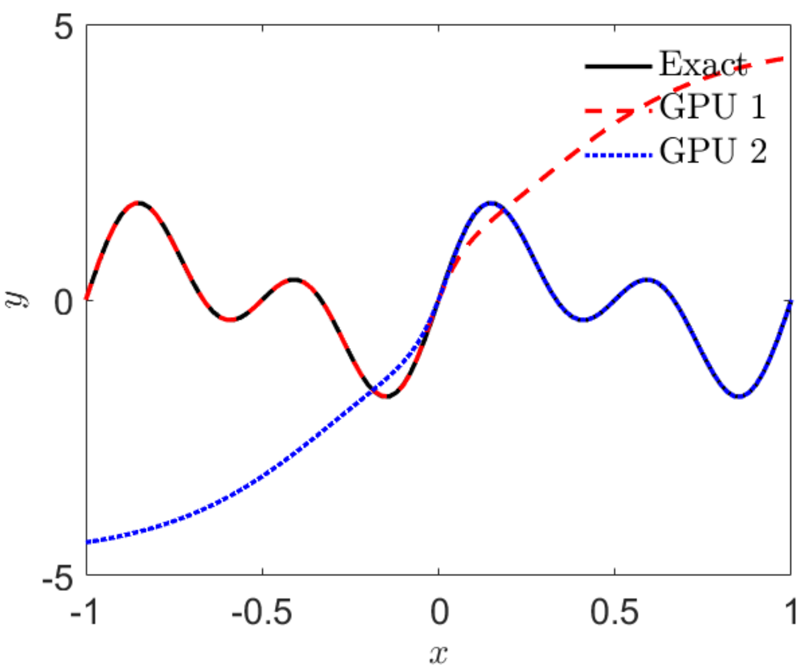
$N = 16$



$depth \times width = 16 \times 2$

GPU 1 $x_{pred} = linspace(-1, 1, 101)$

GPU 2 $x_{pred} = linspace(-1, 1, 101)$



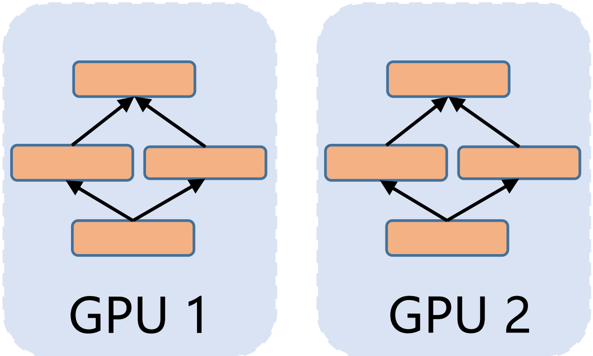
Example 3: PINNs (Data parallelism)

$$-\partial_{xx} u = f, x \in [-1,1]$$

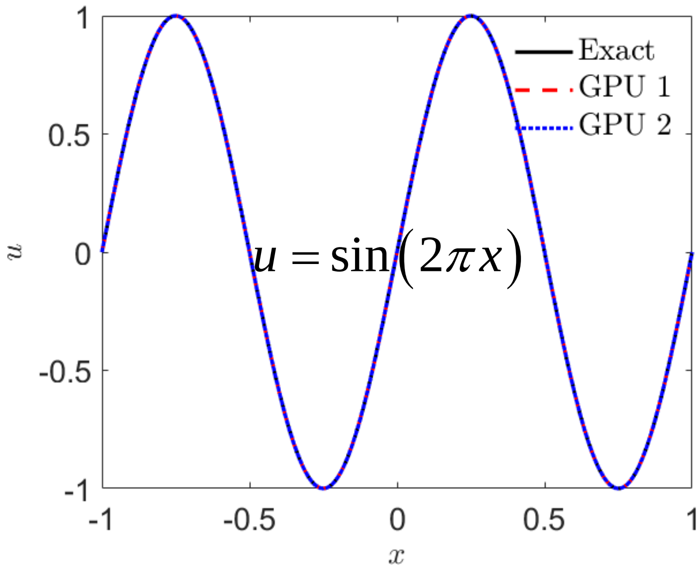
$$f = 4\pi^2 \sin(2\pi x)$$

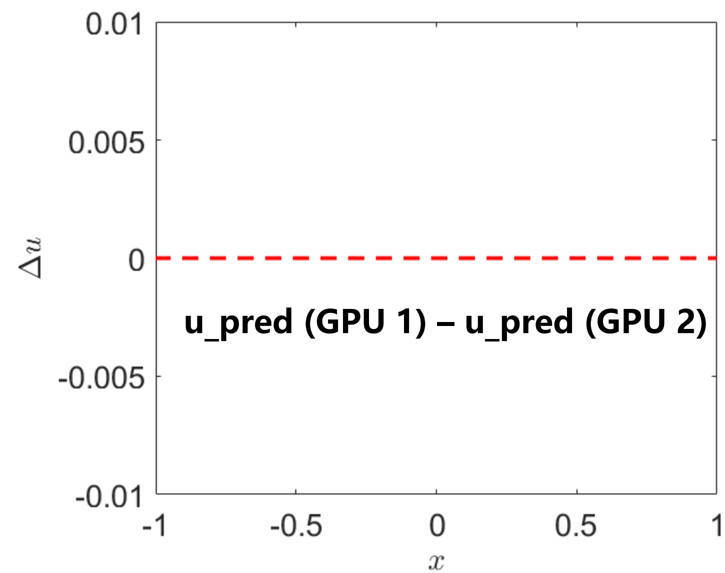
GPU 1 $x = linspace(-1, 0, 51)$

GPU 2 $x = linspace(0, 1, 51)$



$depth \times width = 16 \times 2$





GPU 1 $x = linspace(-1, 1, 201)$

GPU 2 $x = linspace(-1, 1, 201)$

Example 4: PINNs (Ensemble training)

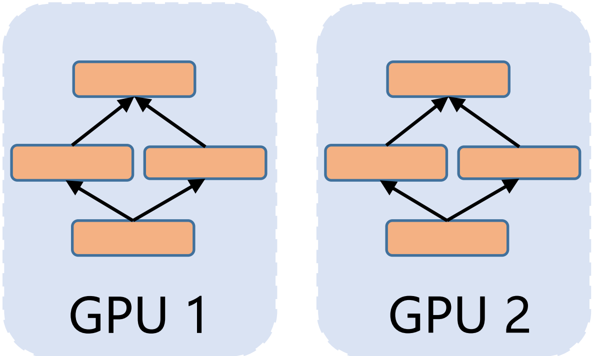
$$-\partial_{xx}u = f, x \in [-1,1]$$

GPU 1 $\quad f = \pi^2 \sin(\pi x)$

GPU 2 $\quad f = 4\pi^2 \sin(2\pi x)$

GPU 1 $\quad u = \sin(\pi x)$

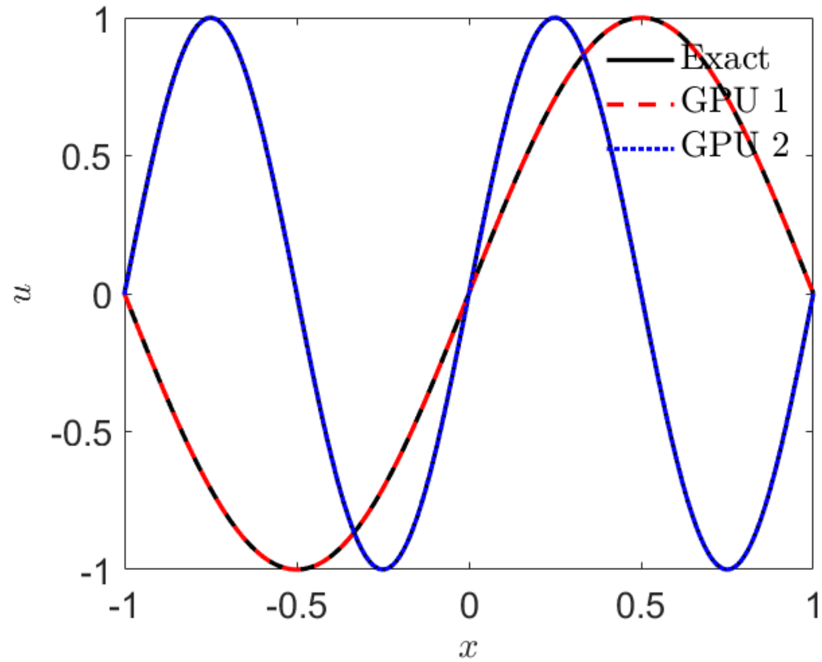
GPU 2 $\quad u = \sin(2\pi x)$



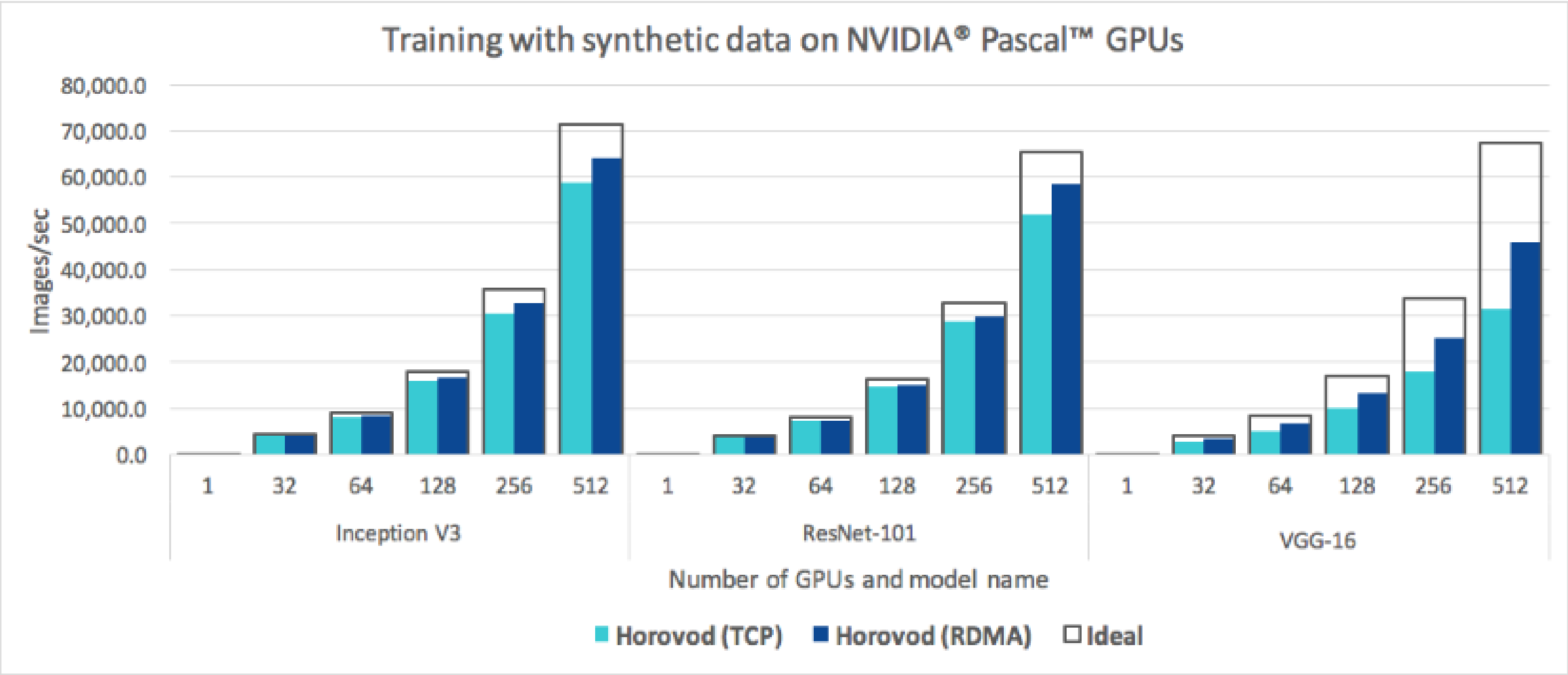
$$depth \times width = 16 \times 2$$

GPU 1 $\quad x = linspace(-1,1,101)$

GPU 2 $\quad x = linspace(-1,1,101)$



Scaling efficiency



The above benchmark was done on 128 servers with 4 Pascal GPUs each connected by a RoCE-capable 25 Gbit/s network. Horovod achieves 90% scaling efficiency for both Inception V3 and ResNet-101, and 68% scaling efficiency for VGG-16.

Installation & Code

- Installation
  https://github.com/horovod/horovod

CCV: Oscar
module load horovod/0.16

- Code download
  https://github.com/XuhuiM/Distributed-training-Horovod.git

Thank you!