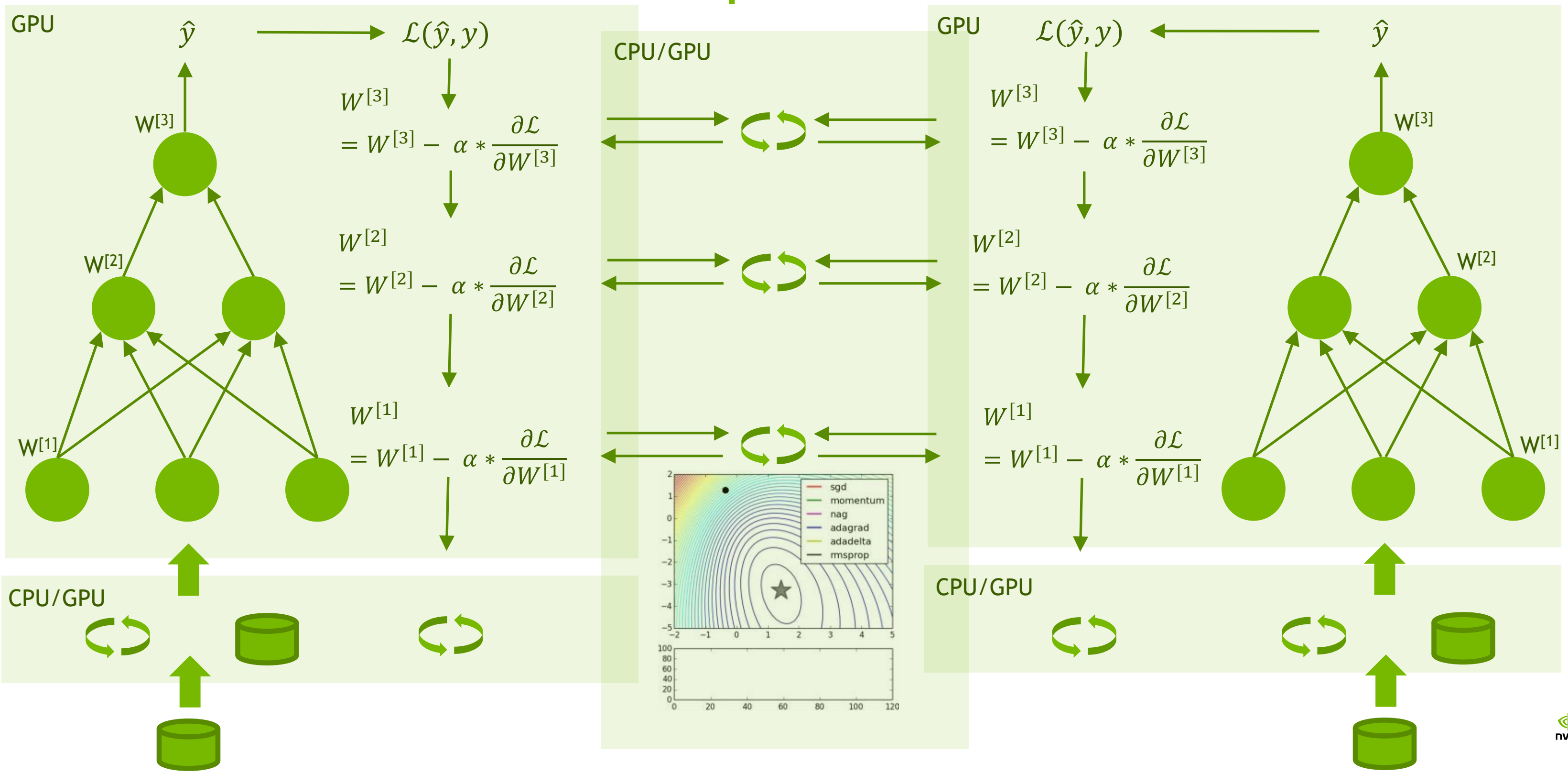
FUNDAMENTALS OF DEEP LEARNING FOR MULTI-GPUS

LAB 2, PART 1: INTRODUCTION TO HOROVOD

NVIDIA | DEEP LEARNING INSTITUTE

TRAINING A NEURAL NETWORK

Multiple GPUs



DEEP LEARNING INSTITUTE

MEET HOROVOD

Library for distributed DL

Works with stock TensorFlow, Keras, PyTorch, and MXNet

Installs with pip

Uses advanced algorithms; leverages high-performance networks (RDMA, GPUDirect).

horovod.ai

MEET HOROVOD

Infrastructure team provides container and MPI environment

ML engineers use DL frameworks that they love

Both have consistent expectations for distributed training across frameworks

horovod.ai

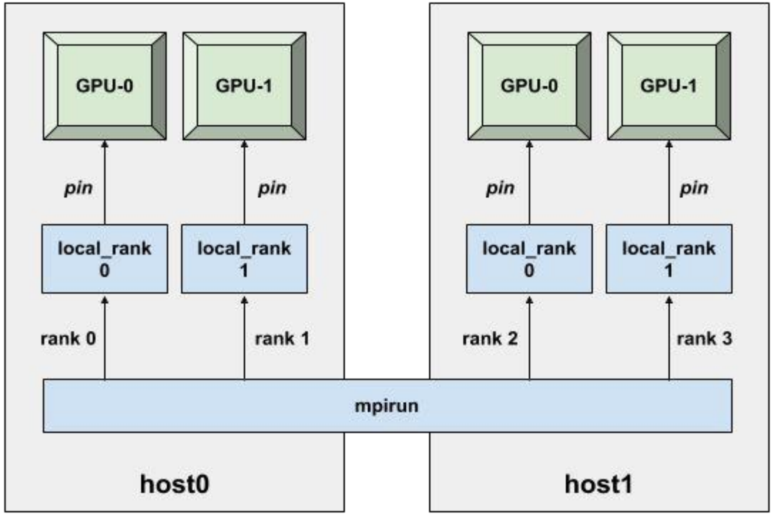
USING HOROVOD

INITIALIZE THE LIBRARY

```
import horovod.tensorflow as hvd

hvd.init()
```

PIN GPU TO BE USED

```
config = tf.ConfigProto()

config.gpu_options.visible_device_list = str(hvd.local_rank())
```

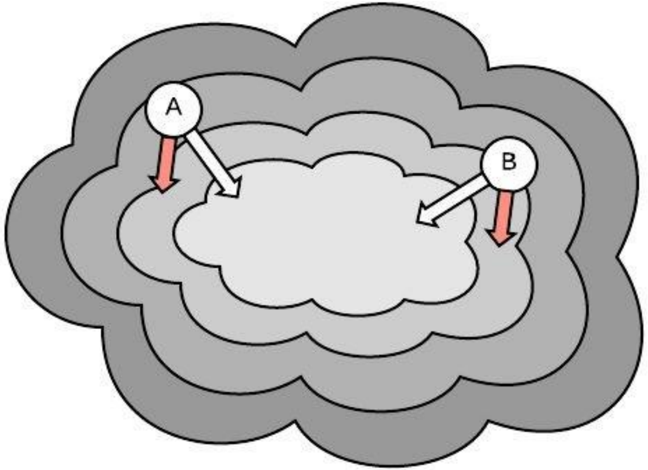


ADD DISTRIBUTED OPTIMIZER

```
opt = hvd.DistributedOptimizer(opt)
```

SYNCHRONIZE INITIAL STATE

```
hooks = [hvd.BroadcastGlobalVariablesHook(0)]

with tf.train.MonitoredTrainingSession(hooks=hooks, …) as sess:

        ...

# Or

bcast_op = hvd.broadcast_global_variables(0)

sess.run(bcast_op)
```



CHECKPOINT ONLY ON ONE WORKER

```
ckpt_dir = "/tmp/train_logs" if hvd.rank() == 0 else None

with tf.train.MonitoredTrainingSession(checkpoint_dir=ckpt_dir, …)
as sess:

...
```

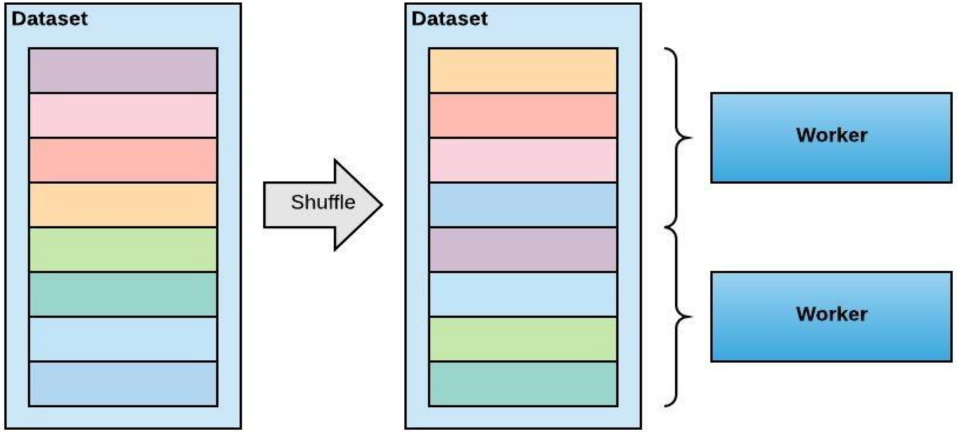
DATA PARTITIONING: OPTION 1

Shuffle the dataset

Partition records among workers

Train by sequentially reading the partition

After epoch is done, reshuffle and partition again



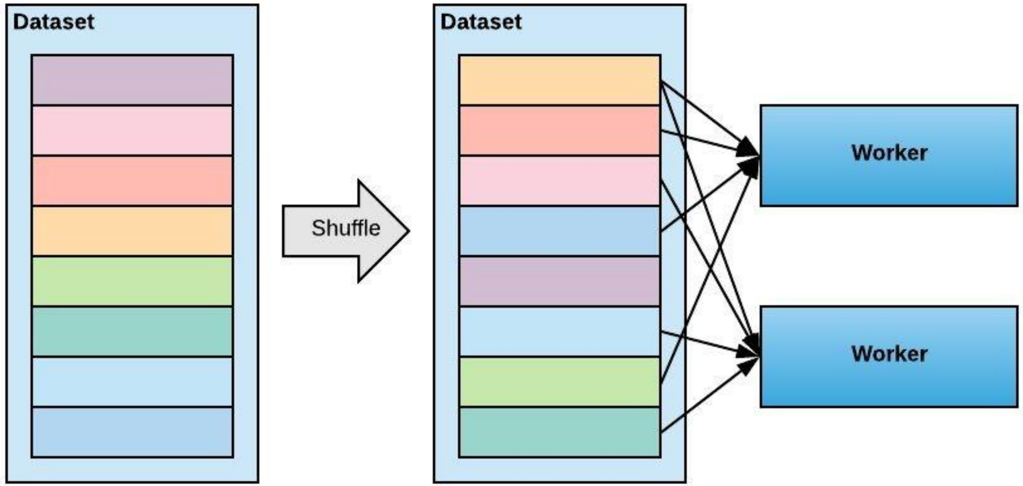
NOTE: make sure that all partitions contain the same number of batches, otherwise the training will deadlock

DATA PARTITIONING: OPTION 2

Shuffle the dataset

Train by randomly reading data from whole dataset

After epoch is done, reshuffle



FULL EXAMPLE IN TENSORFLOW

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


# Initialize Horovod
hvd.init()

# Pin GPU to be used
config = tf.ConfigProto()
config.gpu_options.visible_device_list =
  str(hvd.local_rank())

# Build model...
loss = ...
opt = tf.train.MomentumOptimizer(
  lr=0.01 * hvd.size())

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

# Add hook to synchronize initial state
hooks =[hvd.BroadcastGlobalVariablesHook(0)]

# Only checkpoint on rank 0
ckpt_dir = "/tmp/train_logs" \
  if hvd.rank() == 0 else None

# Make training operation
train_op = opt.minimize(loss)

# The MonitoredTrainingSession takes care of
# session initialization, restoring from a
# checkpoint, saving to a checkpoint, and
# closing when done or an error occurs.
with
tf.train.MonitoredTrainingSession(checkpoint_dir=ckpt_
dir, config=config, hooks=hooks) as mon_sess:
  while not mon_sess.should_stop():
    # Perform synchronous training.
    mon_sess.run(train_op)
```

HOROVOD FOR ALL

```
import horovod.tensorflow as hvd
import horovod.keras as hvd
import horovod.tensorflow.keras as hvd
import horovod.torch as hvd
import horovod.mxnet as hvd
# more frameworks coming
```

RUNNING HOROVOD

Single-node:

```
$ horovodrun -np 4 python train.py
```

Multi-node:

```
$ horovodrun -np 16 -H server1:4,server2:4,server3:4,server4:4
python train.py
```

HOROVOD: UNDER THE HOOD

Run on 4 machines with 4 GPUs:

```
$ mpirun -np 16 \
    -H server1:4,server2:4,server3:4,server4:4 \
    -bind-to none -map-by slot \
    -mca pml ob1 -mca btl ^openib -mca btl_tcp_if_include eth0 \
    -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=eth0 -x
    LD_LIBRARY_PATH -x ... \
    python train.py
```

DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli