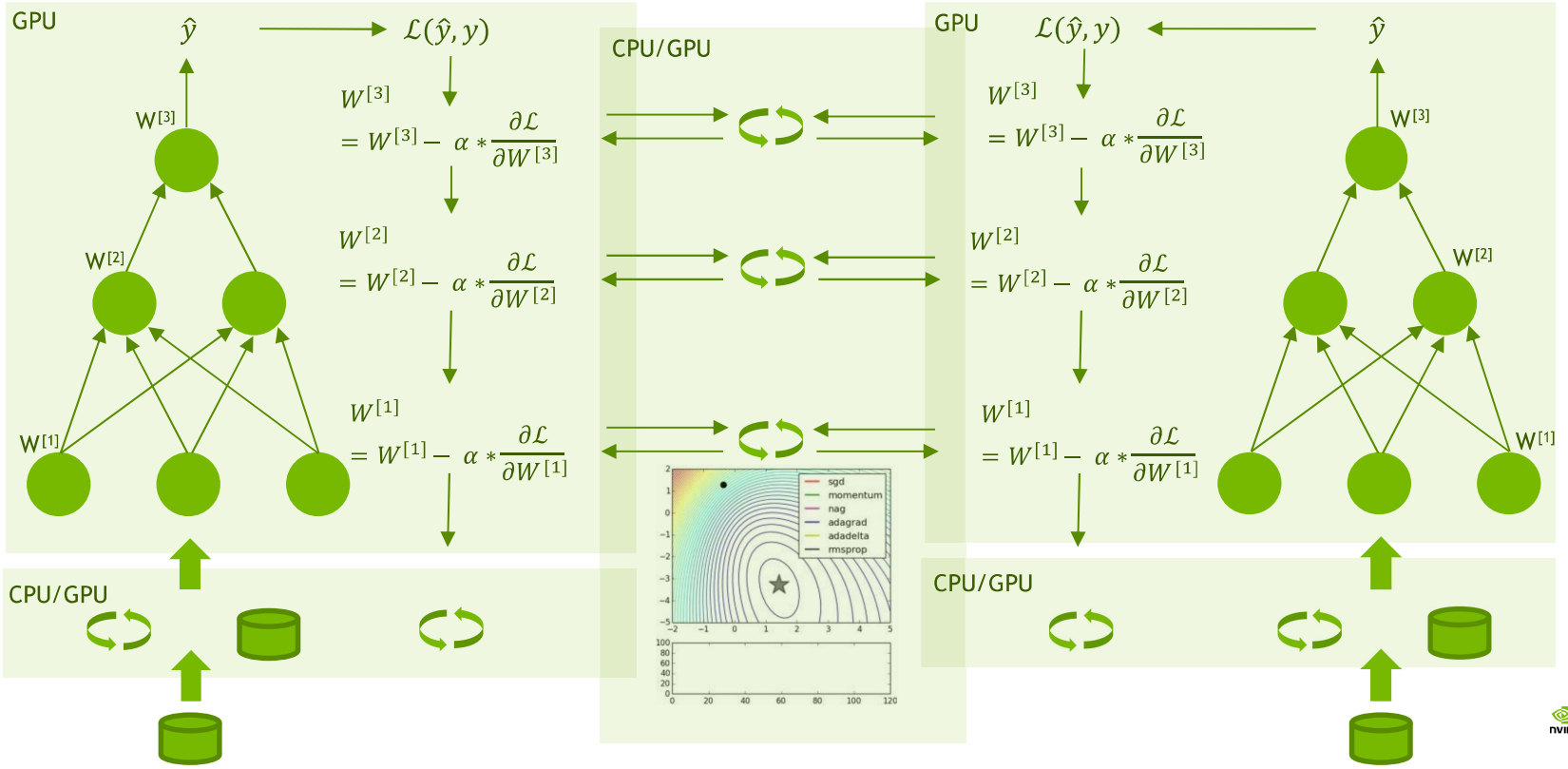
# DATA PARALLELISM: HOW TO TRAIN DEEP LEARNING MODELS ON MULTIPLE GPUS

## LAB 2, PART 1: INTRODUCTION TO DISTRIBUTED DATA PARALLEL (DDP)

# TRAINING A NEURAL NETWORK

## Multiple GPUs



# MEET DDP

Library for distributed DL

Prepackaged into and optimized for PyTorch, an increasingly popular platform among ML engineers and researchers

# USING DISTRIBUTED DATA PARALLEL (DDP)

# INITIALIZE THE PROCESS

```
def setup(global_rank, world_size):
        dist.init_process_group(backend="nccl", rank=global_rank,
        world_size=world_size)
```

# PIN GPU TO BE USED

```
device = torch.device("cuda:" + str(local_rank))
model = Net().to(device)
```

# ENCAPSULATE MODEL WITH DDP

```
model = nn.parallel.DistributedDataParallel(model,
device_ids=[local_rank])
```

# SYNCHRONIZE INITIAL STATE

Handled internally by DDP across processes and nodes!

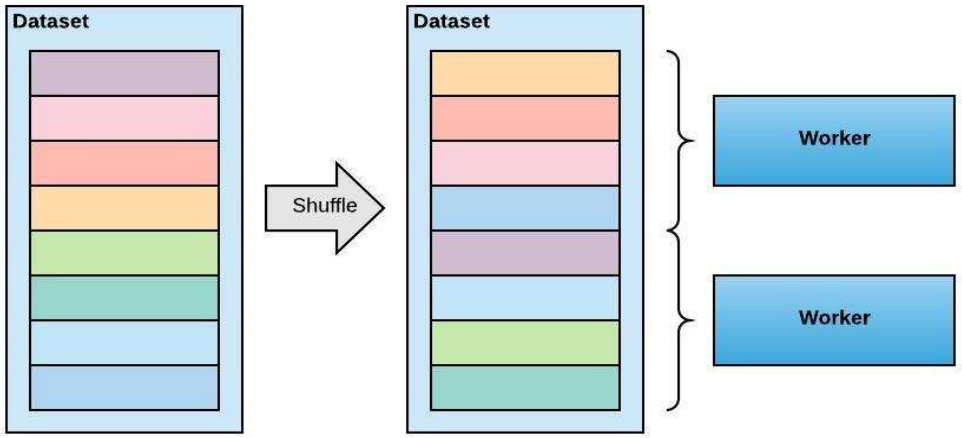
# DATA PARTITIONING

Shuffle the dataset

Partition records among workers

Train by sequentially reading the partition

After epoch is done, reshuffle and partition again



# DATA PARTITIONING

```
train_sampler =
torch.utils.data.distributed.DistributedSampler(train_set,
num_replicas=world_size, rank=global_rank)

train_loader =
torch.utils.data.DataLoader(train_set,
batch_size=args.batch_size, sampler=train_sampler)
```

# I/O ON ONLY ON ONE WORKER

```
download = True if local_rank == 0 else False
if local_rank == 0:
       train_set = torchvision.datasets.FashionMNIST("./data",
download=download)

--------------------

if global_rank == 0:
       print("Epoch = {:2d}: Validation Loss = {:5.3f},
       Validation Accuracy = {:5.3f}".format(epoch+1, v_loss,
       val_accuracy[-1]))
```


www.nvidia.com/dli