IIT Kharagpur | IIT Madras | IIT Goa | IIT Palakkad

Applied Accelerated AI

Scale training on multiple GPUs with PyTorch

Dr. Satyajit Das
Assistant Professor
Data Science
Computer Science and Engineering
IIT Palakkad

National Supercomputing Mission

Centre for Development of Advanced Computing

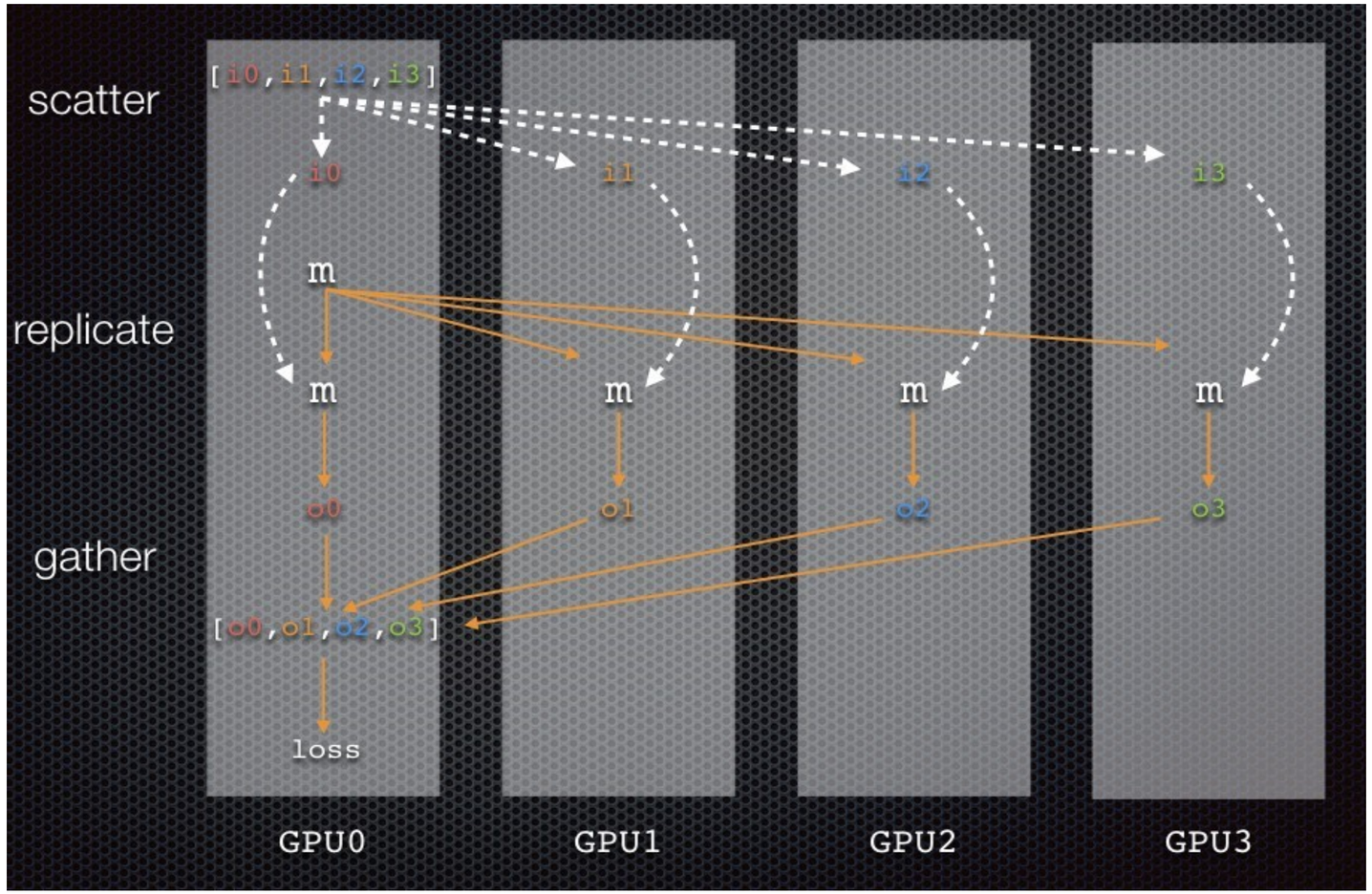
Scale training to multiple GPUs

- Data Parallel – Data distributed across devices
- Model Parallel – Model distributed across devices

Scale training to multiple GPUs

- Single Machine Data Parallel
- Single Machine Model Parallel
- Distributed Data Parallel
- Distributed Data Parallel with Model Parallel
- Distributed Model Parallel

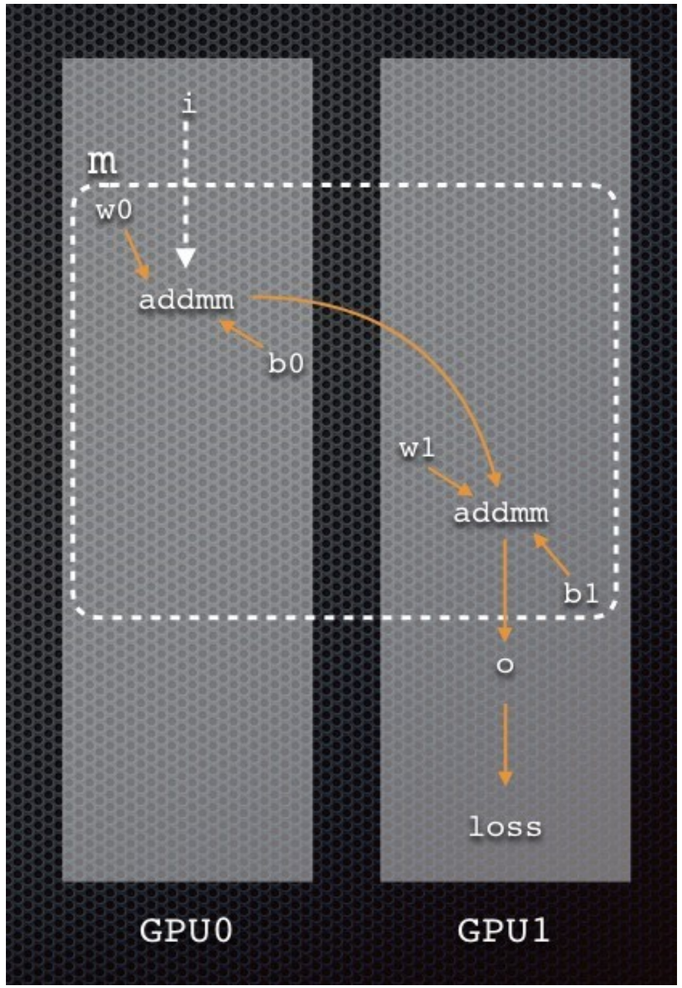
Single Machine Data Parallel




```
model = Net().to("cuda:0")
model = torch.nn.DataParallel(model)

# training loop ...
```

Credit: Vincent Quenneville-Bélair, PhD. Facebook AI.

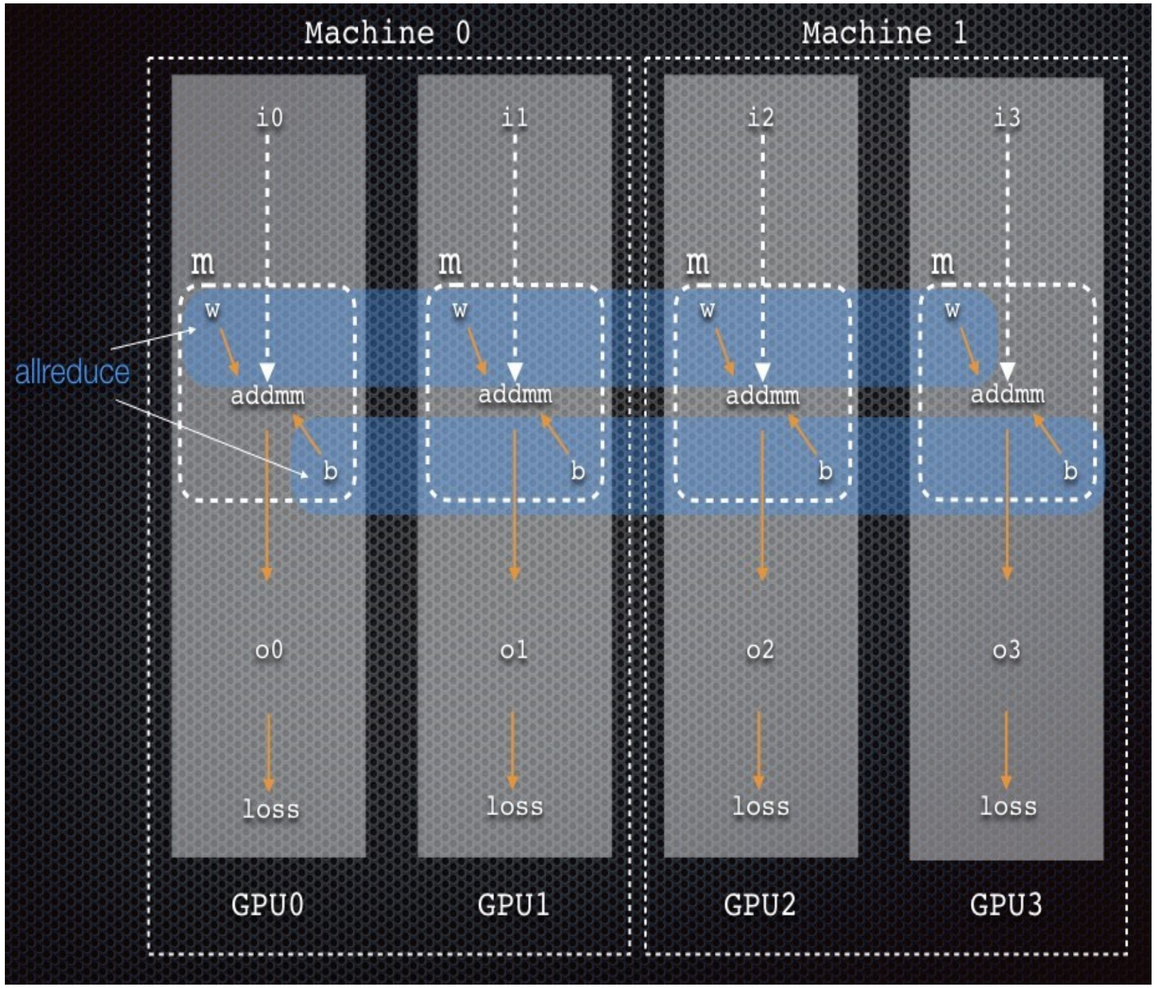
Single Machine Model Parallel



Credit: Vincent Quenneville-Bélair, PhD. Facebook AI.

```
class Net(torch.nn.Module):

  def _init_(self, *gpus):  super(Net)._init
    (self)

    self.gpu0 = torch.device(gpus[0])
    self.gpu1 = torch.device(gpus[1])

    self.sub_net1 = torch.nn.Linear(10, 10).to(self.gpu0)
    self.sub_net2 = torch.nn.Linear(10, 5).to(self.gpu1)


  def forward(self, x):
    y = self.sub_net1(x.to(self.gpu0))
    z = self.sub_net2(y.to(self.gpu1))  # blocking
    return z


model = Net("cuda:0", "cuda:1")


# training loop...
```

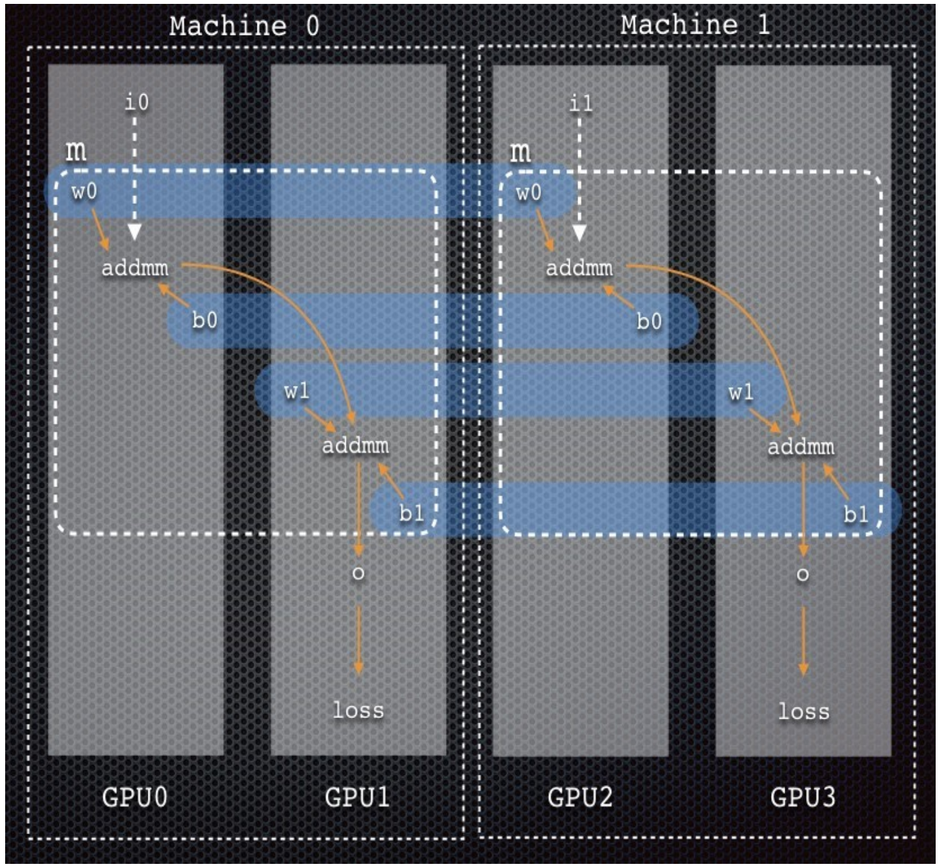
Distributed Data Parallel



Credit: Vincent Quenneville-Bélair, PhD. Facebook AI.

```
def one_machine(machine_rank, world_size,
  backend):  torch.distributed.init_process_group(
    backend, rank=machine_rank, world_size=world_size
  )


  gpus = {
    0: [0, 1],
    1: [2, 3],
  }[machine_rank]  # or one gpu per process to avoid GIL


  model = Net().to(gpus[0])  # default to first gpu on
  machine
  model = torch.nn.parallel.DDP(model, device_ids=gpus)


  # training loop...

for machine_rank in range(world_size):
  torch.multiprocessing.spawn(
    one_machine, args=(world_size, backend),
    nprocs=world_size, join=True # blocking
  )
```

Distributed Data Parallel with Model Parallel




```
def one_machine(machine_rank, world_size, backend):
  torch.distributed.init_process_group(
    backend, rank=machine_rank, world_size=world_size
  )


  gpus = {0: [0, 1],
          1: [2, 3],}[machine_rank]


  model = Net(gpus)
  model = torch.nn.parallel.DDP(model)


  # training loop...

for machine_rank in range(world_size):
  torch.multiprocessing.spawn(
    one_machine, args=(world_size, backend),
    nprocs=world_size, join=True
  )
```

Credit: Vincent Quenneville-Bélair, PhD. Facebook AI.

Thank You