Pytorch Training

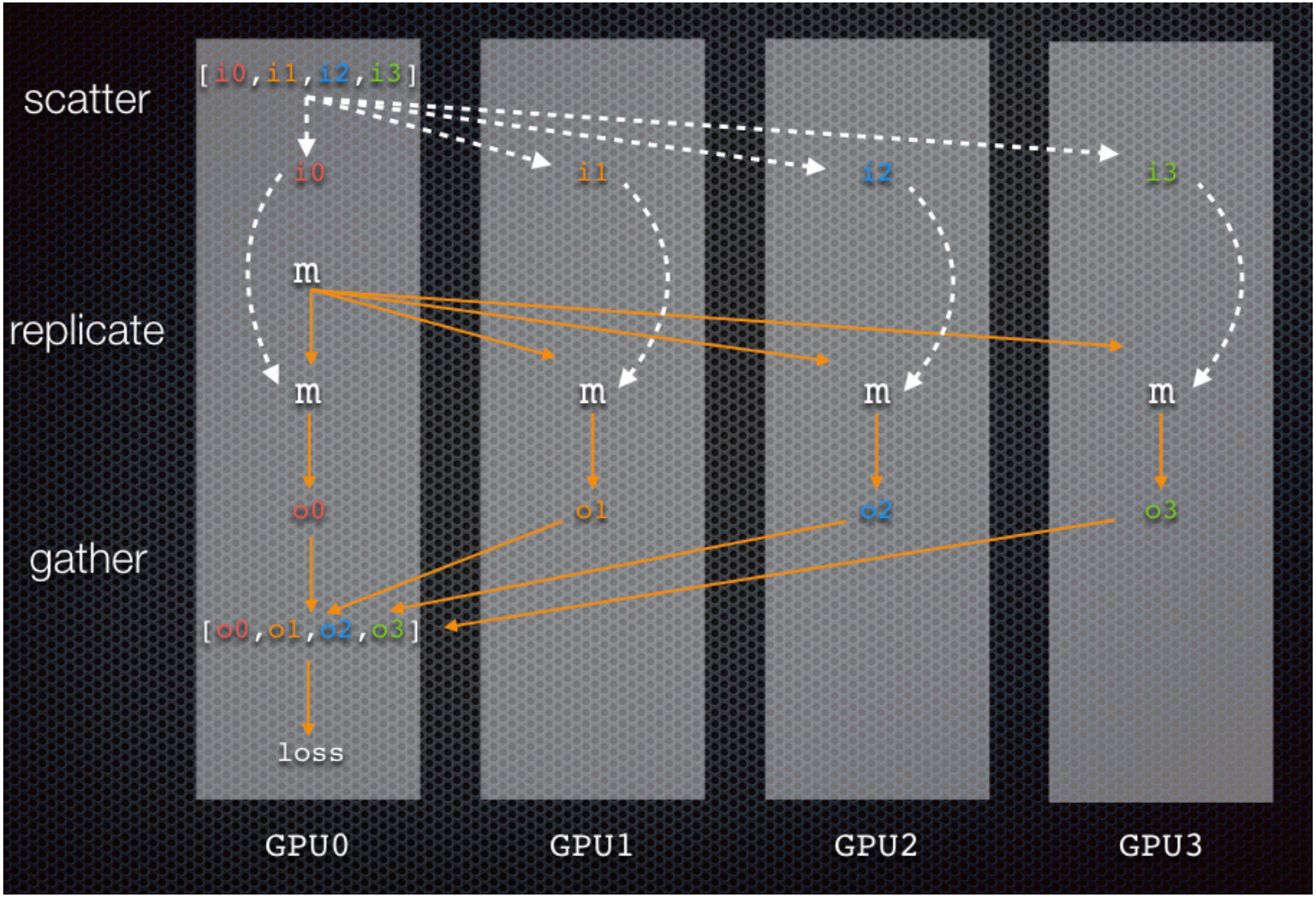
Use Multiple Gpus and Machines

Data Parallel – Data distributed across devices

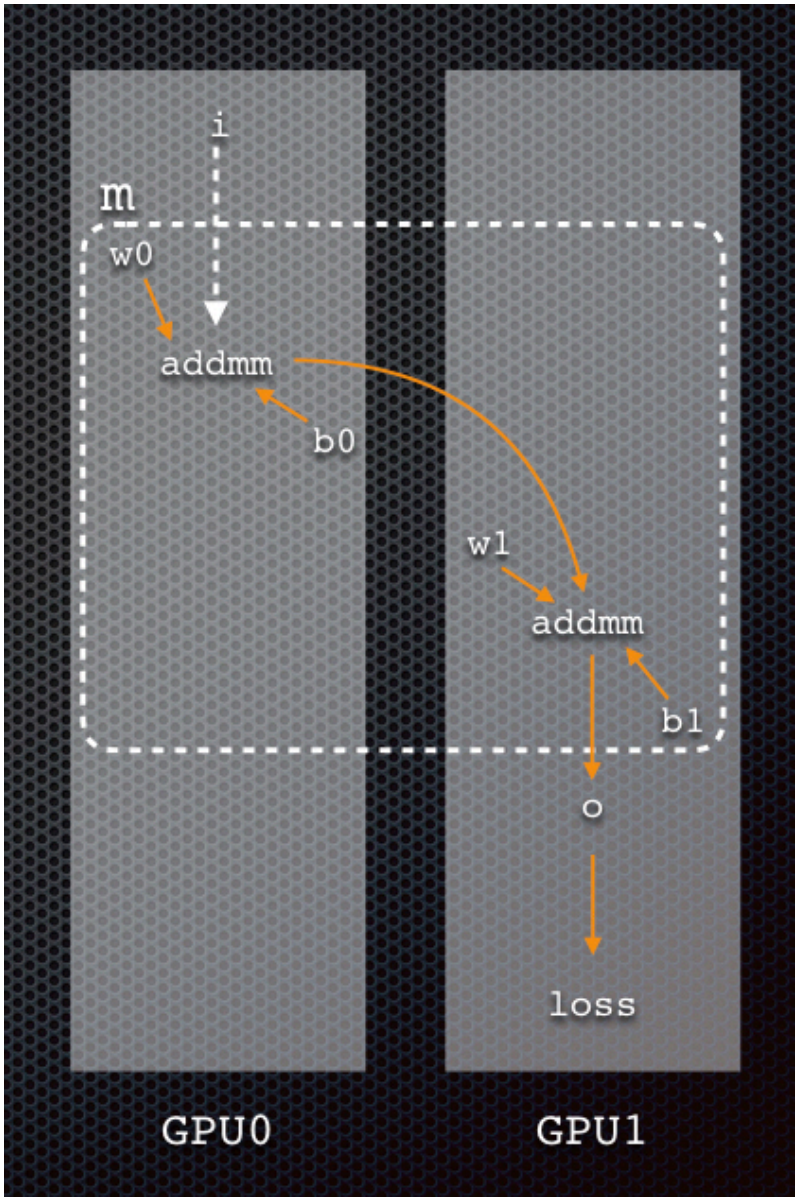
Model Parallel – Model distributed across devices

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

Single Machine Data Parallel



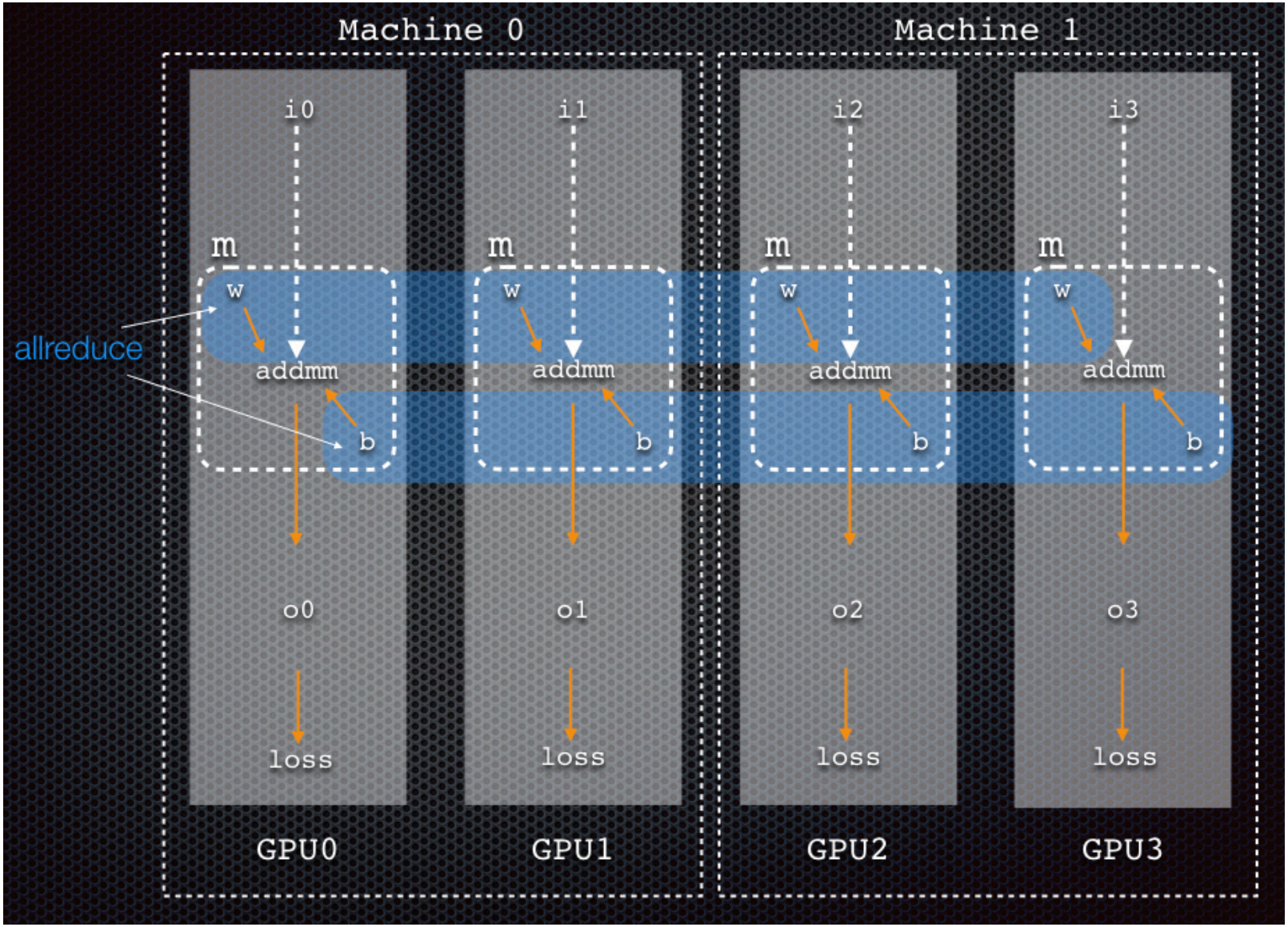
Single Machine Model Parallel



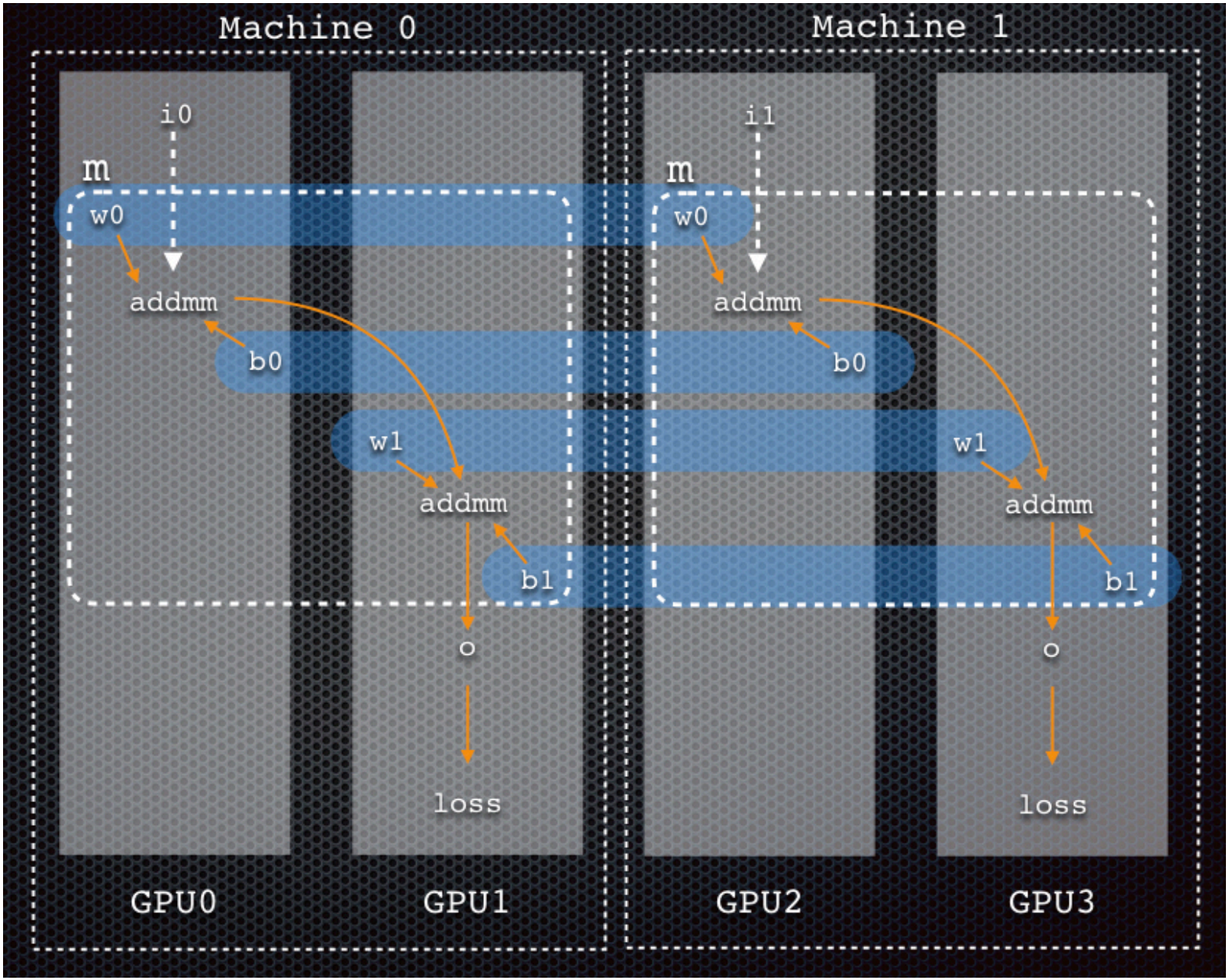

```
class Net(torch.nn.Module):

    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

https://pytorch.org/tutorials/intermediate/dist_tuto.html#collective-communication



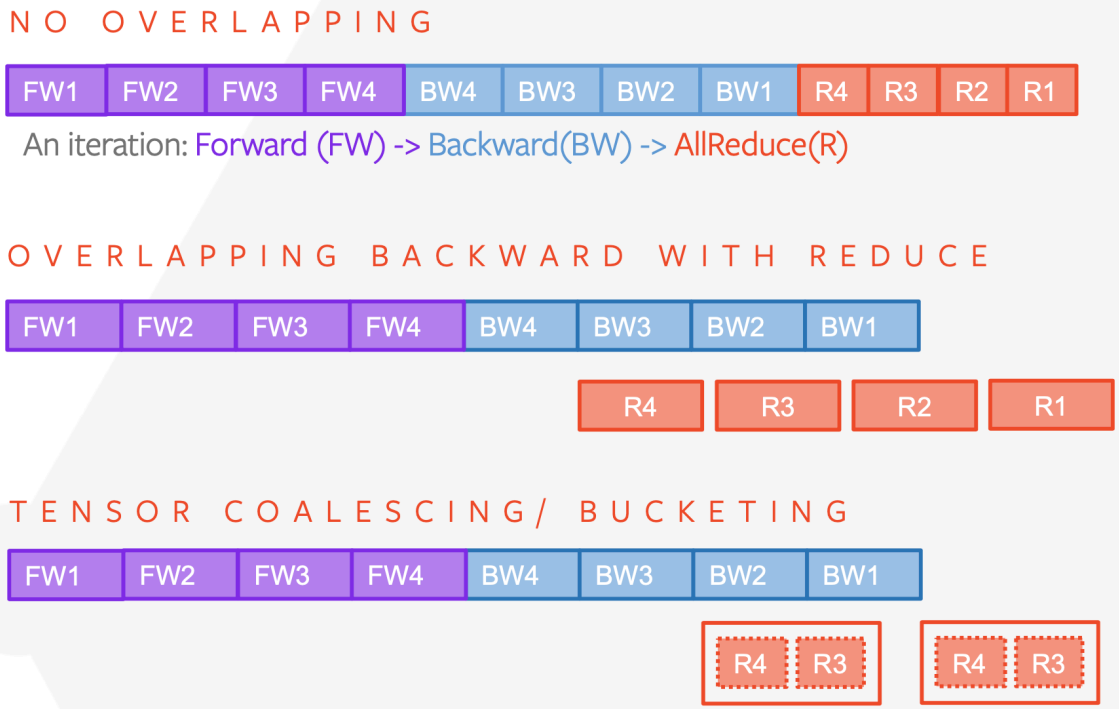
Distributed Data Parallel with Model Parallel

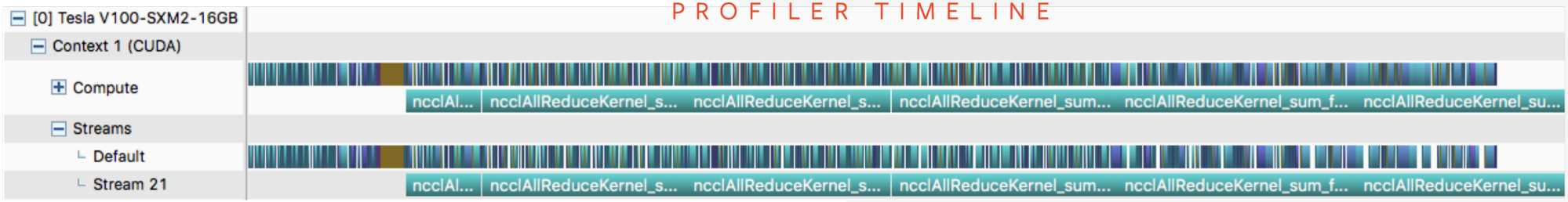


DISTRIBUTED DATA PARALLEL

Performance-driven design

- Overlapping BWs with all-reductions
- Coalescing small tensors into buckets
  - A bucket is a big coalesced tensor





https://pytorch.org/tutorials/intermediate/ddp_tutorial.html