Distributed Data Parallel Training

Distributed Stochastic Gradient Descent

- core operation: global sum reduction (already seen this)
- other issues:
  - arrange for initialization with same weights
  - arrange for different kinds of training data for each node

In [1]:

```
# make sure we are starting with a clean slate
kubectl delete jobs --all
kubectl delete pods --all
```

```
No resources found
No resources found
```

Configuration

- headless service enables DNS resolution in cluster
- communications on port 9000
- environment variables tell clients where to connect

In [2]:

```
cat > kubetpl.yaml <<'EOF'
image: gcr.io/research-191823/bigdata19
memory: 4G
cpu: 1
app: bigdata19
subdomain: bigdata19
port:
  - 9000
config_map: files
env:
  - MASTER_ADDR=master.bigdata19
  - MASTER_PORT=9000
EOF
```

In [3]:

```
kubectl delete service/bigdata19 || true
kubetpl service | kubectl apply -f -
```

```
service "bigdata19" deleted
service/bigdata19 created
```

Few Changes Needed for Distributed Training

In [4]:

```
diff training.py disttraining.py || true
```

```
10a11
> import torch.distributed as dist
25a27
> parser.add_argument("-P", "--sample-probability", type=float, default=1.0)
26a29,30
> parser.add_argument("--seed", type=int, default=902842093840)
> parser.add_argument("--dist", default="-1/-1")
32a37,41
> rank, world = [int(x) for x in args.dist.split("/")]
>
> if world > 0:
>     dist.init_process_group("gloo", rank=rank, world_size=world)
>
61a71,73
> if world > 0:
>     torch.manual_seed(args.seed)
>
68a81,84
>
> if world > 0:
>     model = nn.parallel.DistributedDataParallel(model)
>     torch.manual_seed(args.seed+173434*rank)
```

In [5]:

```
kubefcm files reduce.py disttraining.py training.py helpers.py
```

```
-- --from-file=reduce.py=reduce.py
-- --from-file=disttraining.py=disttraining.py
-- --from-file=training.py=training.py
-- --from-file=helpers.py=helpers.py
configmap "files" deleted
configmap/files created
```

Distributed Training

- arrange for the same weights in each node by setting the same seed
  - explicit distribution of weights might be better
  - alternative: load same starting network in all of them
- arrange for different training data in each node
  - here we rely on different random shuffling in `WebDataset`
  - PyTorch examples use explicit splitting (not necessary)

Create the Master and Compute Nodes

In [ ]:

```
kubectl delete pods --all || true
kubetpl pod -n master -G 1 -c 'cp /files/*.py .; python3 disttraining.py --dist 0/4' | kubectl apply -f -
for i in {1..3}; do
    kubetpl pod -n node$i -G 1 -c "cp /files/*.py .; python3 disttraining.py --dist $i/4" | kubectl apply -f -
done
```

In [12]:

```
sleep 30
```

In [13]:

```
kubectl get pods
```

```
NAME     READY   STATUS    RESTARTS   AGE
master   1/1     Running   0          49s
node1    1/1     Running   0          32s
node2    1/1     Running   0          31s
node3    1/1     Running   0          31s
```

In [16]:

```
kubectl logs master
```

```
/opt/conda/lib/python3.6/site-packages/torchvision/io/_video_opt.py:17: UserWarning: video reader ba
sed on ffmpeg c++ ops not available
  warnings.warn("video reader based on ffmpeg c++ ops not available")
Mon Dec  9 20:40:15 UTC 2019; master; root; /workspace; GPU 0: Tesla T4 (UUID: GPU-98cb0870-4950-111
e-9140-5d7ed3a2c273);
creating resnet50
        0 bs   128 per sample loss 5.47e-02 loading 2.06e-02 training 2.30e-02
      512 bs   128 per sample loss 5.50e-02 loading 1.70e-02 training 1.87e-02
     1024 bs   128 per sample loss 5.50e-02 loading 1.46e-02 training 1.59e-02
     1536 bs   128 per sample loss 5.51e-02 loading 1.31e-02 training 1.40e-02
     2048 bs   128 per sample loss 5.51e-02 loading 1.21e-02 training 1.27e-02
     2560 bs   128 per sample loss 5.51e-02 loading 1.14e-02 training 1.19e-02
     3072 bs   128 per sample loss 5.50e-02 loading 1.09e-02 training 1.16e-02
     3584 bs   128 per sample loss 5.48e-02 loading 1.06e-02 training 1.13e-02
     4096 bs   128 per sample loss 5.46e-02 loading 1.05e-02 training 1.11e-02
     4608 bs   128 per sample loss 5.45e-02 loading 1.03e-02 training 1.11e-02
     5120 bs   128 per sample loss 5.45e-02 loading 1.03e-02 training 1.10e-02
     5632 bs   128 per sample loss 5.45e-02 loading 1.03e-02 training 1.10e-02
     6144 bs   128 per sample loss 5.44e-02 loading 1.02e-02 training 1.09e-02
     6656 bs   128 per sample loss 5.43e-02 loading 1.02e-02 training 1.10e-02
     7168 bs   128 per sample loss 5.43e-02 loading 1.01e-02 training 1.13e-02
     7680 bs   128 per sample loss 5.42e-02 loading 9.95e-03 training 1.15e-02
     8192 bs   128 per sample loss 5.42e-02 loading 9.92e-03 training 1.16e-02
     8704 bs   128 per sample loss 5.42e-02 loading 1.04e-02 training 1.17e-02
     9088 bs   128 per sample loss 5.42e-02 loading 1.03e-02 training 1.27e-02
     9600 bs   128 per sample loss 5.42e-02 loading 1.02e-02 training 1.25e-02
```

In [17]:

```
kubectl logs node3
```

```
/opt/conda/lib/python3.6/site-packages/torchvision/io/_video_opt.py:17: UserWarning: video reader ba
sed on ffmpeg c++ ops not available
  warnings.warn("video reader based on ffmpeg c++ ops not available")
Mon Dec  9 20:40:15 UTC 2019; node3; root; /workspace; GPU 0: Tesla T4 (UUID: GPU-fd29201b-d663-6697
-b413-a761dceb23c8);
creating resnet50
        0 bs   128 per sample loss 5.52e-02 loading 2.13e-02 training 2.26e-02
      512 bs   128 per sample loss 5.53e-02 loading 1.75e-02 training 1.85e-02
     1024 bs   128 per sample loss 5.52e-02 loading 1.50e-02 training 1.56e-02
     1536 bs   128 per sample loss 5.52e-02 loading 1.34e-02 training 1.38e-02
     2048 bs   128 per sample loss 5.52e-02 loading 1.23e-02 training 1.26e-02
     2560 bs   128 per sample loss 5.49e-02 loading 1.16e-02 training 1.18e-02
     3072 bs   128 per sample loss 5.48e-02 loading 1.10e-02 training 1.15e-02
     3584 bs   128 per sample loss 5.48e-02 loading 1.07e-02 training 1.12e-02
     4096 bs   128 per sample loss 5.47e-02 loading 1.05e-02 training 1.11e-02
     4608 bs   128 per sample loss 5.46e-02 loading 1.06e-02 training 1.09e-02
     5120 bs   128 per sample loss 5.45e-02 loading 1.04e-02 training 1.09e-02
     5632 bs   128 per sample loss 5.45e-02 loading 1.03e-02 training 1.10e-02
     6144 bs   128 per sample loss 5.45e-02 loading 1.02e-02 training 1.10e-02
     6656 bs   128 per sample loss 5.44e-02 loading 1.04e-02 training 1.09e-02
     7168 bs   128 per sample loss 5.44e-02 loading 1.04e-02 training 1.10e-02
     7680 bs   128 per sample loss 5.43e-02 loading 1.04e-02 training 1.10e-02
     8192 bs   128 per sample loss 5.42e-02 loading 1.05e-02 training 1.10e-02
     8704 bs   128 per sample loss 5.41e-02 loading 1.06e-02 training 1.14e-02
     9088 bs   128 per sample loss 5.42e-02 loading 1.05e-02 training 1.25e-02
     9600 bs   128 per sample loss 5.41e-02 loading 1.05e-02 training 1.23e-02
```

In [18]:

```
kubectl get pods
```

```
NAME     READY   STATUS      RESTARTS   AGE
master   0/1     Completed   0          10m
node1    0/1     Completed   0          10m
node2    0/1     Completed   0          10m
node3    0/1     Completed   0          10m
```

In [ ]:

```

```