Lab 10: Multi-GPU Scaling

Stuart Harley

Introduction: In this lab I will be using multiple GPUs to train my self-driving car model to explore scaling with multiple GPUs. I am training all of the models in the lab with a batch size of 32 run for 10 epochs.

Learning Outcomes:

- Multi-GPU training understanding
- Usage of Horovod
- Scaling projections

Running on 2 GPUs

Shown: Screenshot of GPU utilization

```
Every 0.5s: nvidia-smi

Wed Feb 19 10:31:14 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.33.01    Driver Version: 440.33.01    CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:60:00.0 Off |                    0 |
| N/A   32C    P8     9W /  70W |      0MiB / 15109MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla T4            On   | 00000000:61:00.0 Off |                    0 |
| N/A   49C    P0    28W /  70W |   1166MiB / 15109MiB |     10%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla T4            On   | 00000000:DA:00.0 Off |                    0 |
| N/A   49C    P0    29W /  70W |   1166MiB / 15109MiB |     11%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla T4            On   | 00000000:DB:00.0 Off |                    0 |
| N/A   30C    P8    12W /  70W |      0MiB / 15109MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    1     13326      C   python                                     1155MiB |
|    2     13327      C   python                                     1155MiB |
+-----------------------------------------------------------------------------+
```

Shown: Loss output from running

```
[1,0]<stdout>:Epoch 10/10
[1,1]<stdout>: - 46s - loss: 0.0199 - accuracy: 0.1812 - val_loss: 0.0158 - val_accuracy: 0.1801
[1,1]<stdout>:Epoch 10/10
[1,0]<stdout>: - 44s - loss: 0.0190 - accuracy: 0.1849 - val_loss: 0.0226 - val_accuracy: 0.1718
[1,0]<stdout>:dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
[1,0]<stdout>:Loss
[1,0]<stdout>:[1.116979814917836, 0.17555272284226064, 0.05216957034777529, 0.02796803237316597, 0.02221885413572883, 0.02113728017434524, 0.020168618132539142, 0.019983836262916554, 0.019960274771842736, 0.019023630032765 4]
[1,0]<stdout>:Validation loss
[1,0]<stdout>:[0.34776949882507324, 0.08860749006271362, 0.036292508244514465, 0.032170750200748444, 0.015620336878689, 0.020992930978536606, 0.014610119163990 02, 0.013355077244341373, 0.018431508913636208, 0.022590898399972 9]
```

```
[1,1]<stdout>: - 44s - loss: 0.0189 - accuracy: 0.1781 - val_loss: 0.0122 - val_accuracy: 0.1887
[1,1]<stdout>:dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
[1,1]<stdout>:Loss
[1,1]<stdout>:[1.214956333305662, 0.17506621896553381, 0.053153253585988325, 0.02706873706156654, 0.02189424331145768, 0.020922002601177766, 0.019979654219223 57, 0.020038748536238098, 0.019870760745757464, 0.018888524499447 272]
[1,1]<stdout>:Validation loss
[1,1]<stdout>:[0.33120885491371155, 0.08478444814682007, 0.03270806372716568, 0.04341091960668564, 0.013513795100152493, 0.019833967089653015, 0.023300634697079 66, 0.015720708295702934, 0.015788244083523 75, 0.012182897888123989]
```

It took about 8 minutes to train the model. The loss for both GPUs is shown above. It is about .019 for both.

The rest of the jobs ran on their own, so I do not have GPU utilization screenshots for them.

Running on 4 interactive (T4) GPUs

It took about 7.5 minutes to train the model. The loss across the 4 GPUs was about .046.

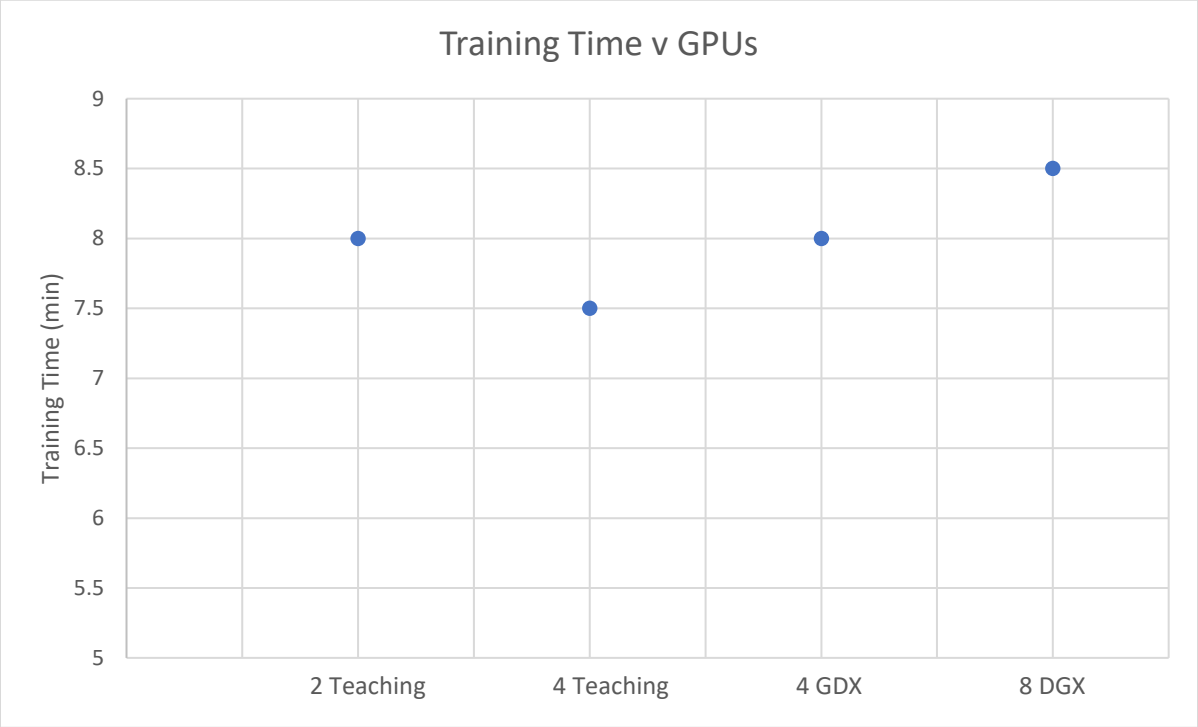
Running on 4 DGX GPUs

It took about 8 minutes to train the model. The loss across the 4 DGX GPUs was about .047.

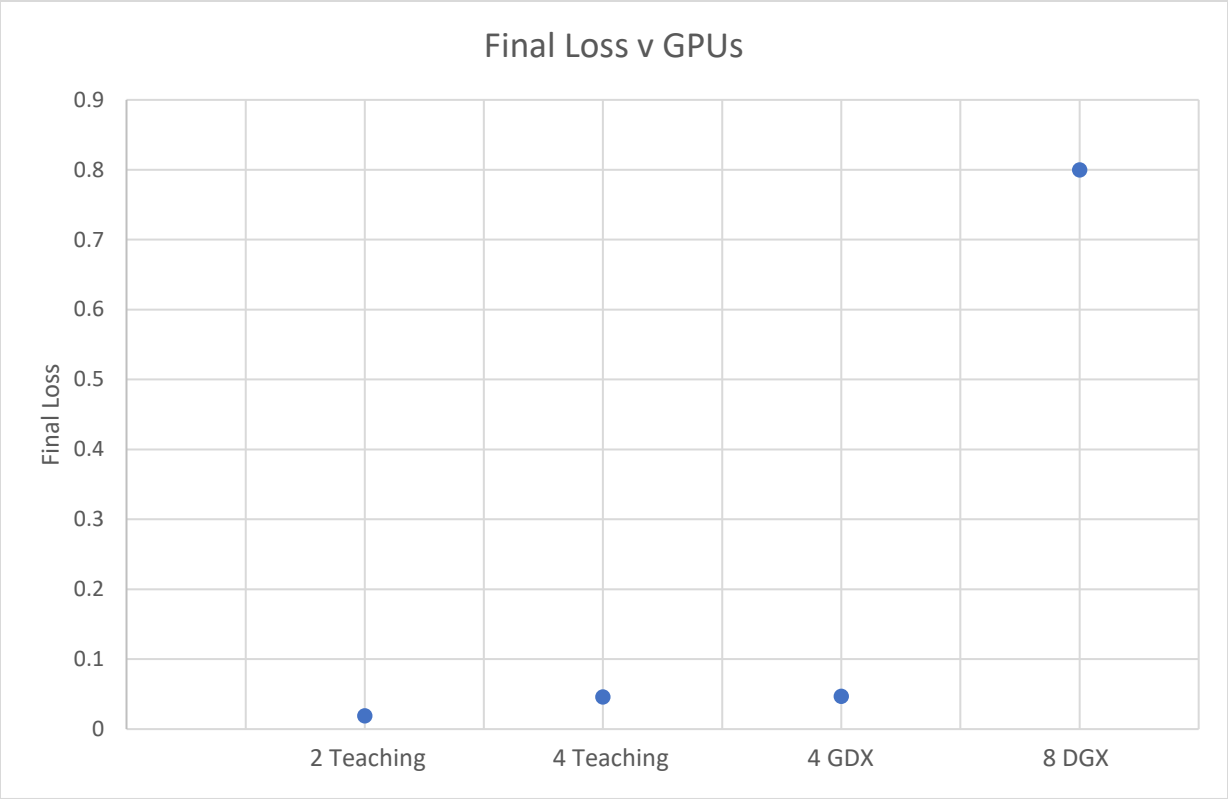
Running on 8 DGX GPUs

It took about 8.5 minutes to train the model. The loss across the 8 DGX GPUs was about .8.

Shown: Plot of the training times for the models



Shown: Plot of the losses for the models



Conclusion: Our dataset is not large enough to utilize multiple GPUs effectively. It is very much so overkill for a dataset of this size. And since using more GPUs slows convergence, really the only thing I found from running these jobs is that the more GPUs that were used, the larger the loss was. There were also minimal differences in running time between the models. This is because the averaging of the model weights between the GPUS after each epoch, offsets the time gains from splitting up the dataset between the different GPUs. Again however, our dataset is too small to effectively use multiple GPUs. if our dataset was larger, we would most likely see time improvements.