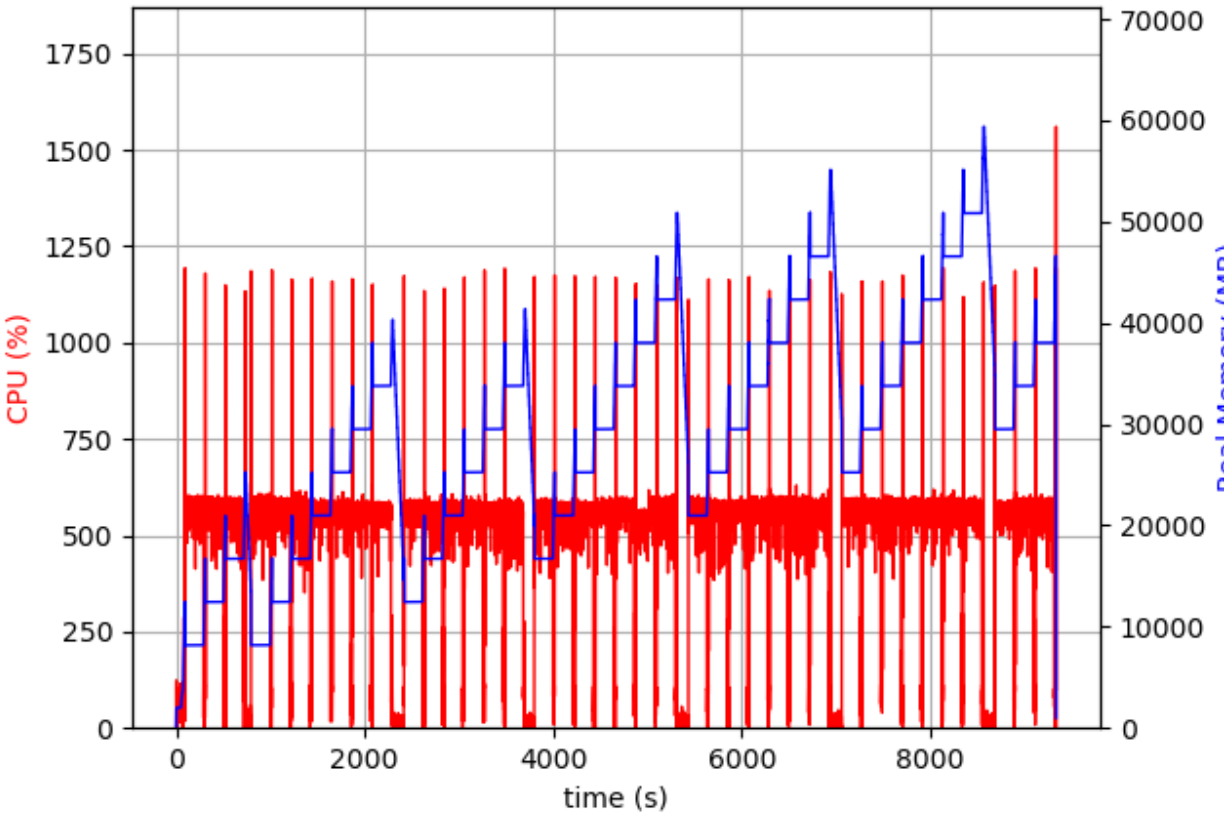
Question 1:

Efficient implementations of all reduce should be faster than parameter server when the hardware is homogeneous. When using heterogeneous hardware, it may be better to use the parameter server.

Question 2:

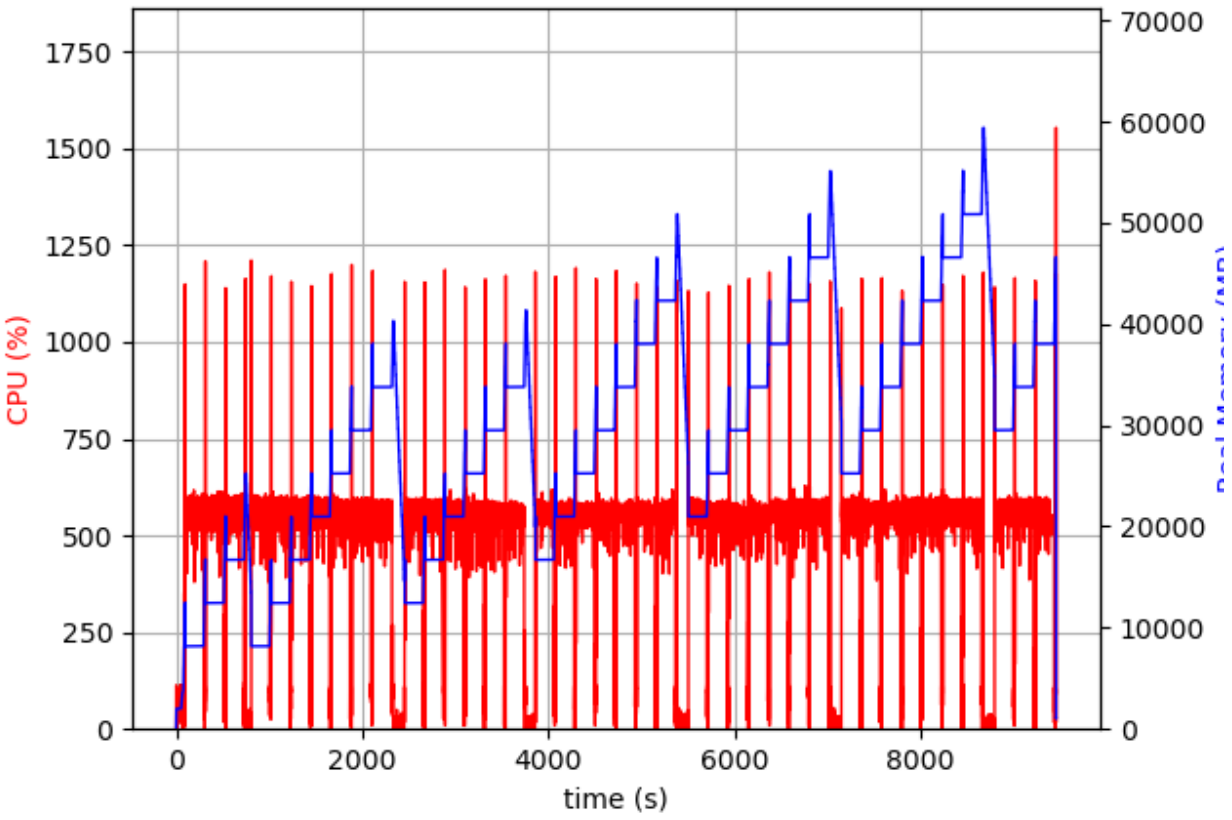
9238.130583047867 seconds to train



CPU utilization is pretty consistent for the entire run. Every change in CPU usage seems to correspond to a change in memory usage. There are drops in CPU usage when the garbage collector frees memory and spikes after the memory is freed.

Question 3:

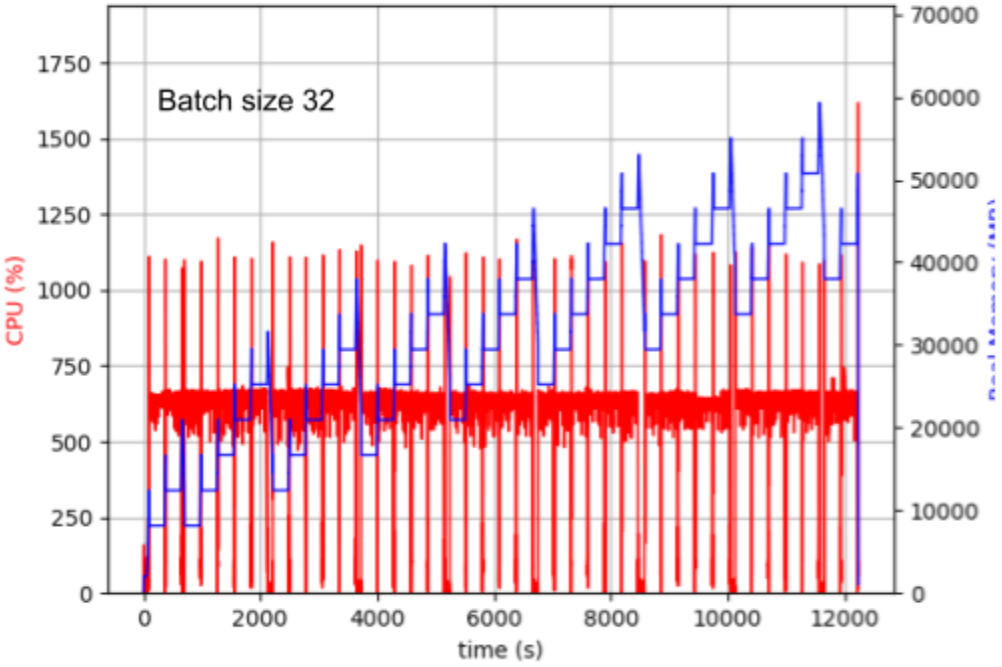
9353.67615866661 seconds to train

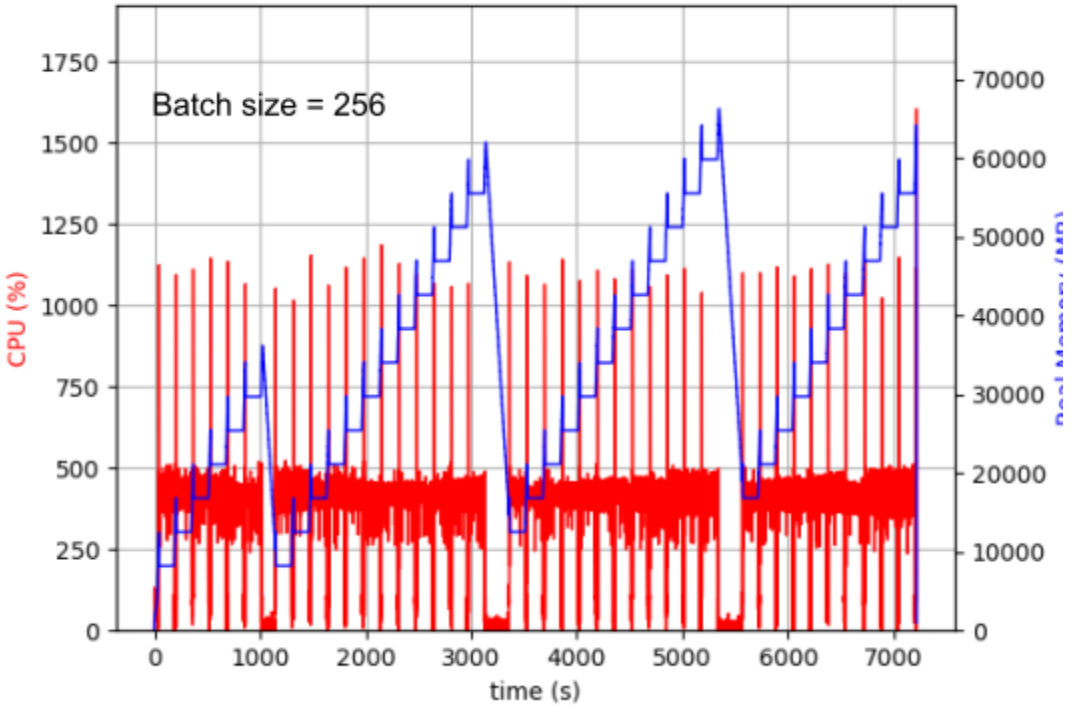


Checkpointing increases runtime, but not by much since only a few checkpoints are run.

Question 4:

batch size 32 : 12142.804160356522 seconds to train
batch size 256 : 7170.198219060898 seconds to train


Batch size 32
CPU (%)
time (s)
0
250
500
750
1000
1250
1500
1750
0
2000
4000
6000
8000
10000
12000
0
10000
20000
30000
40000
50000
60000
70000

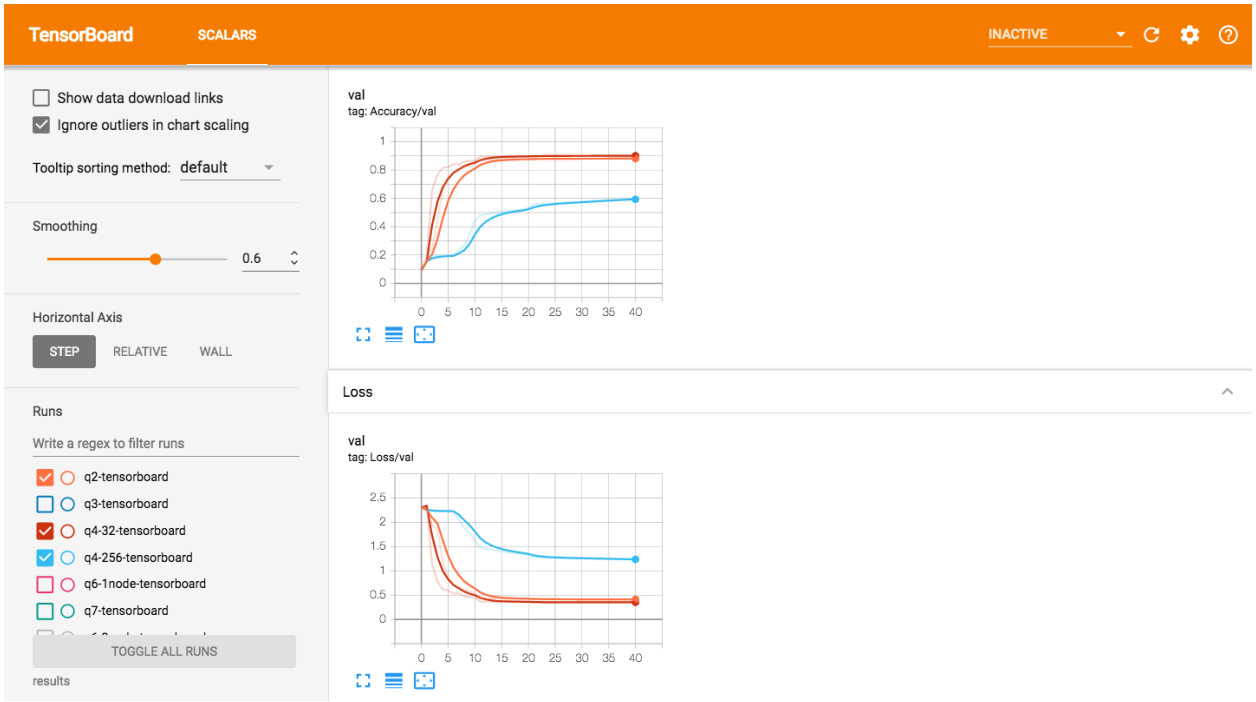

Batch size = 256
CPU (%)
time (s)
0
250
500
750
1000
1250
1500
1750
0
1000
2000
3000
4000
5000
6000
7000
0
10000
20000
30000
40000
50000
60000
70000

Smaller batch sizes have higher CPU utilization, but larger batch sizes finish the 40 epochs more quickly because there are fewer total iterations.

Larger batch sizes gave worse accuracy, this is because there are fewer weight updates.

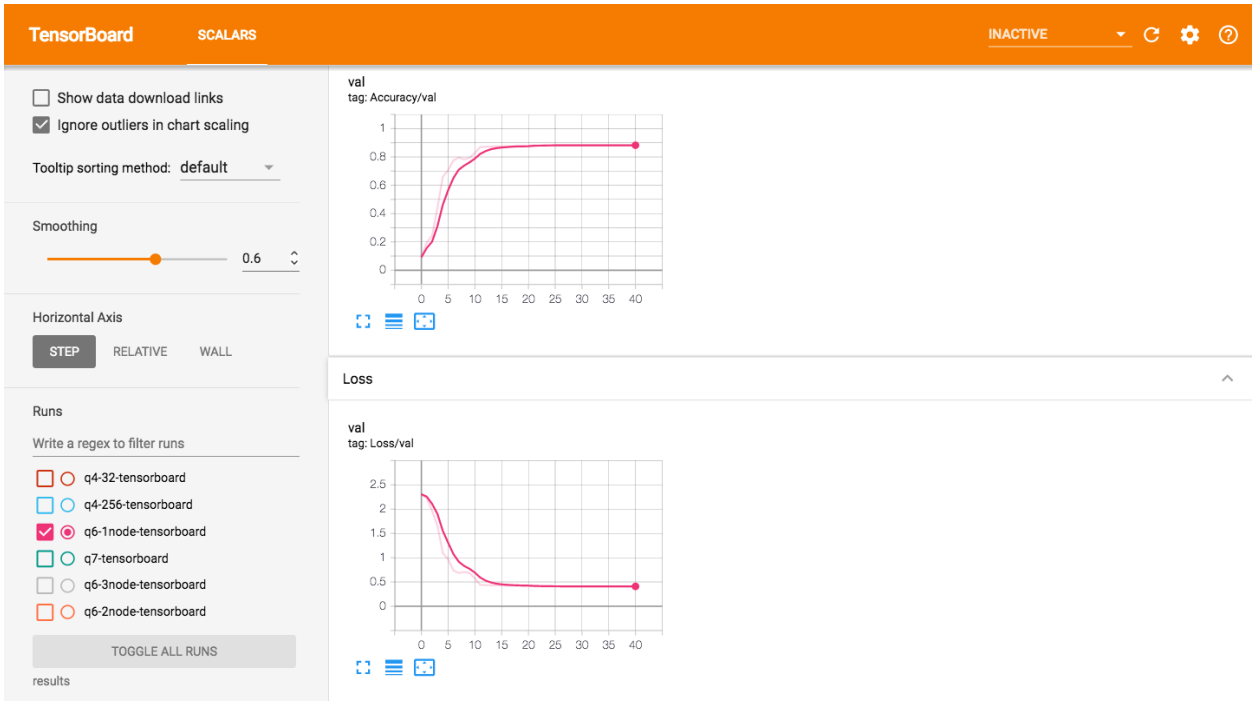
Question 5:

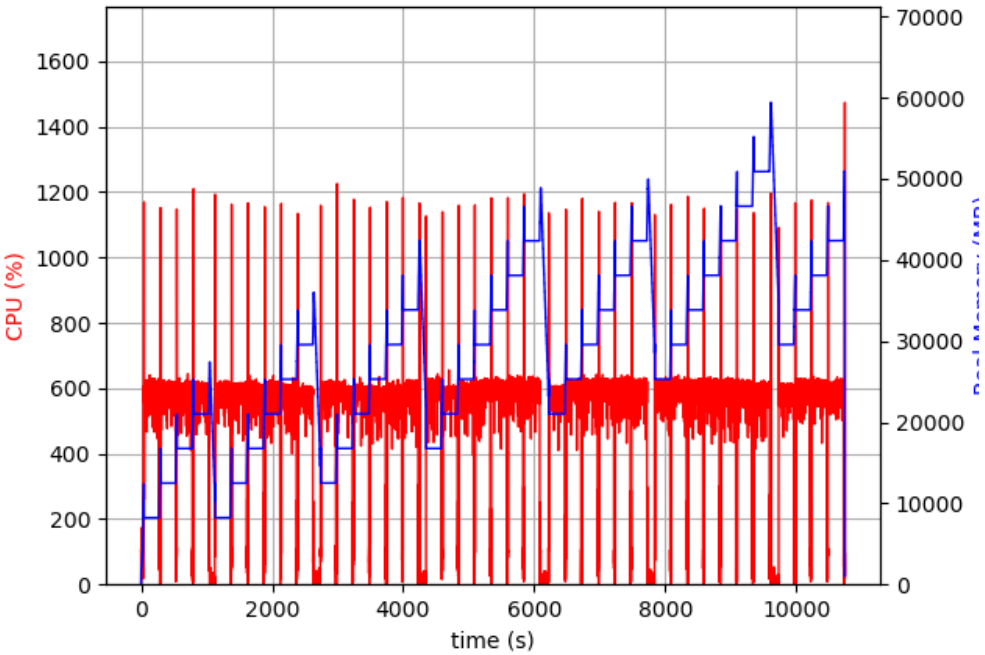
This figure shows tensorflow plots for question 2 and 4. Batch size of 32 trains fastest.



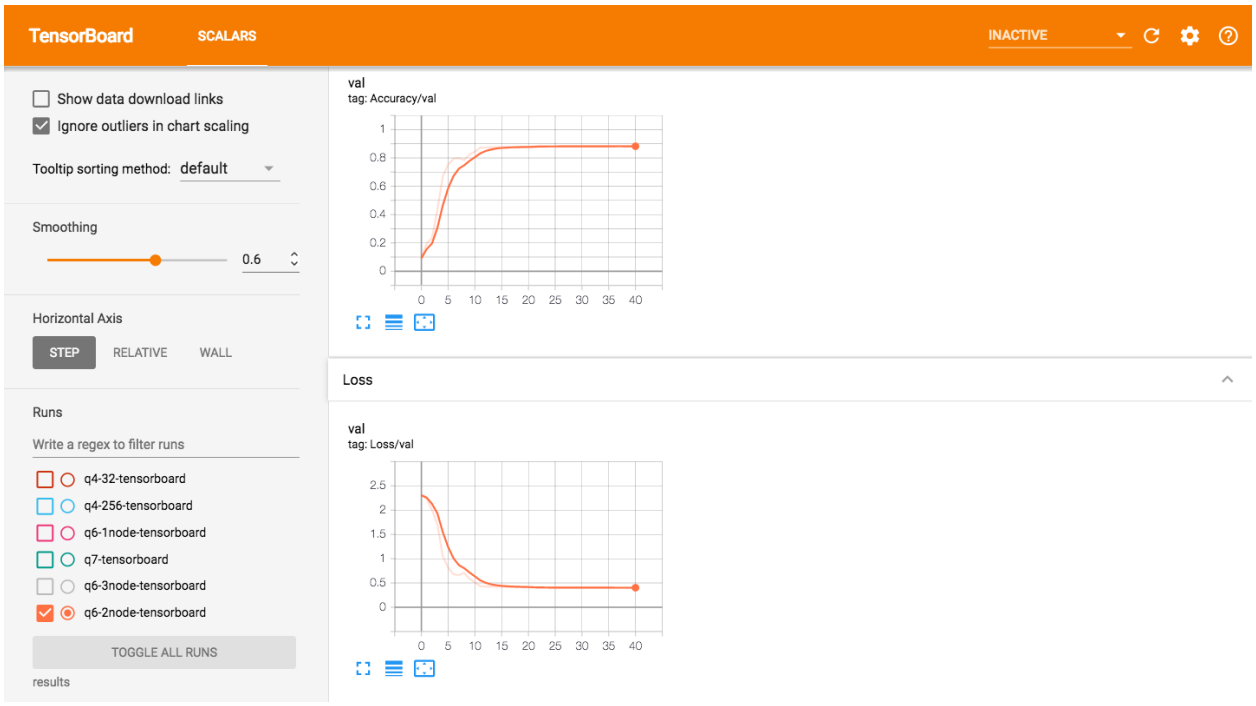
Question 6:

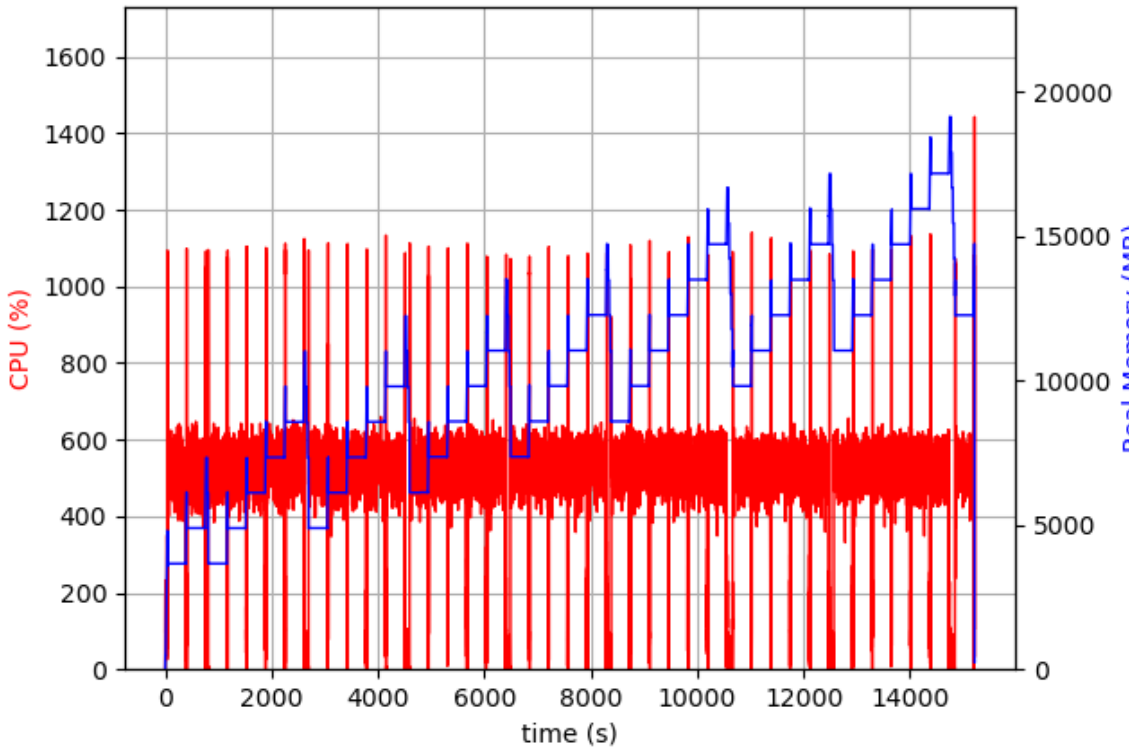
1 node : 10694.454135894775 seconds to train - batch size of 64



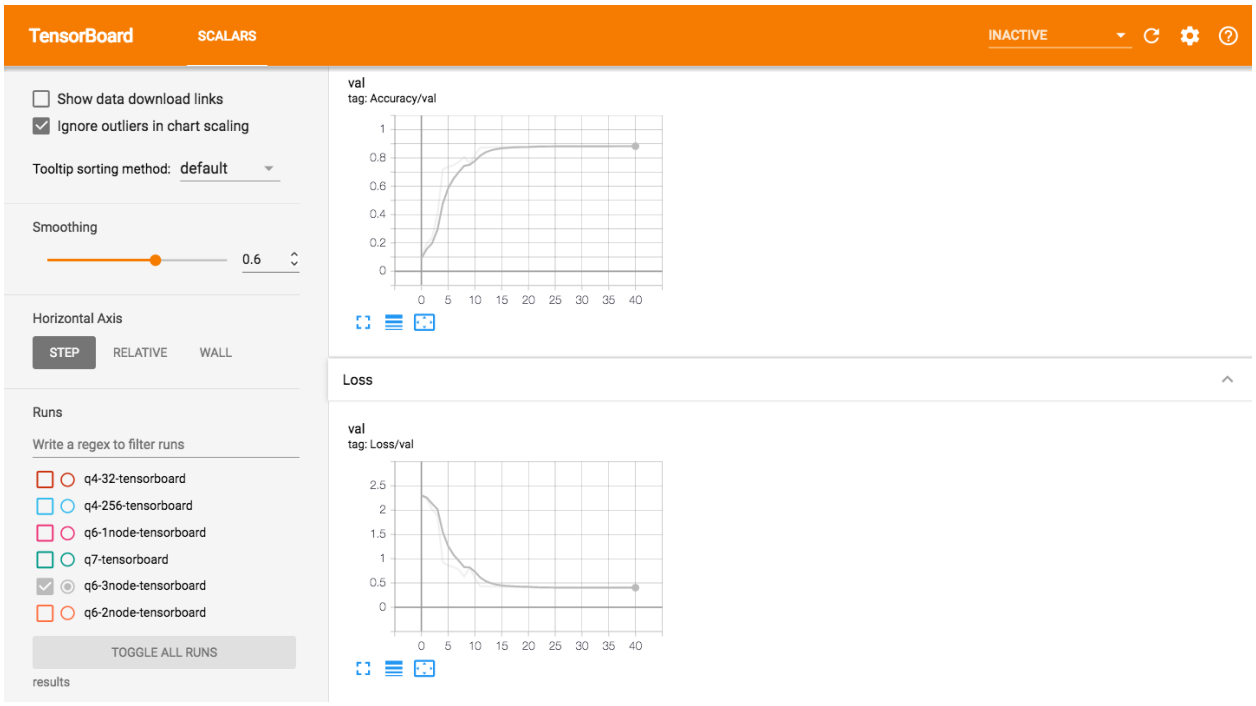


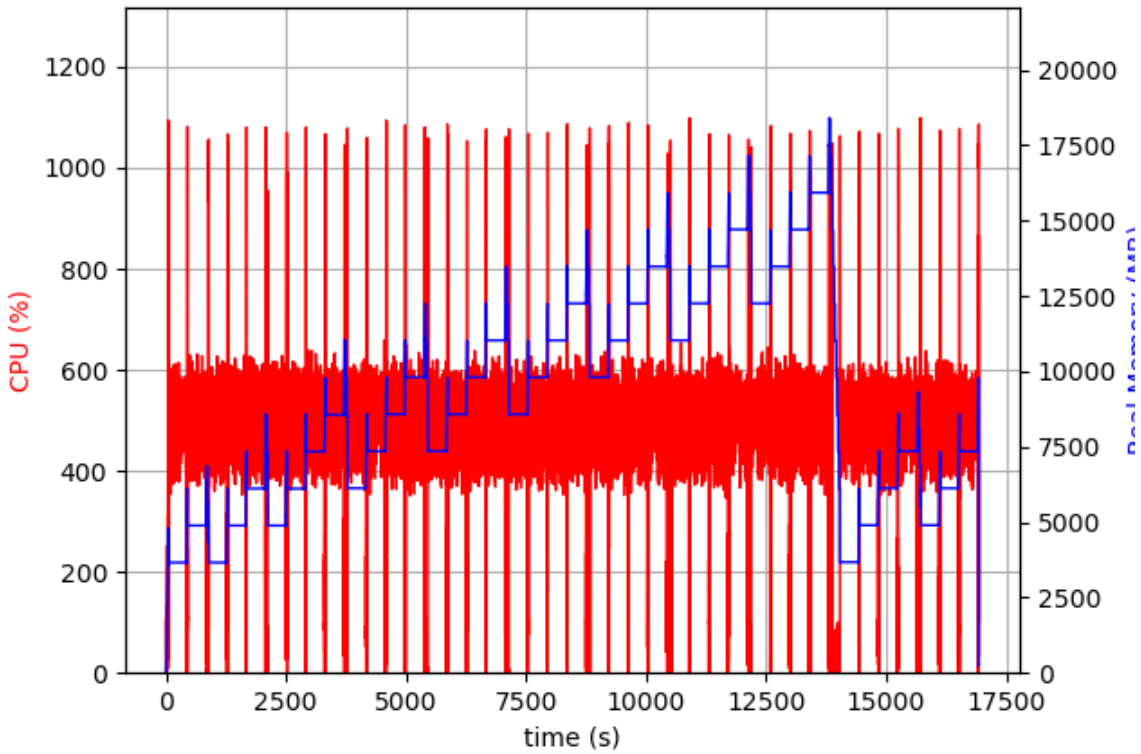
2 node : 15169.921590805054 seconds to train - batch size of 32





3 node : 16879.323309659958 seconds to train - batch size of 21

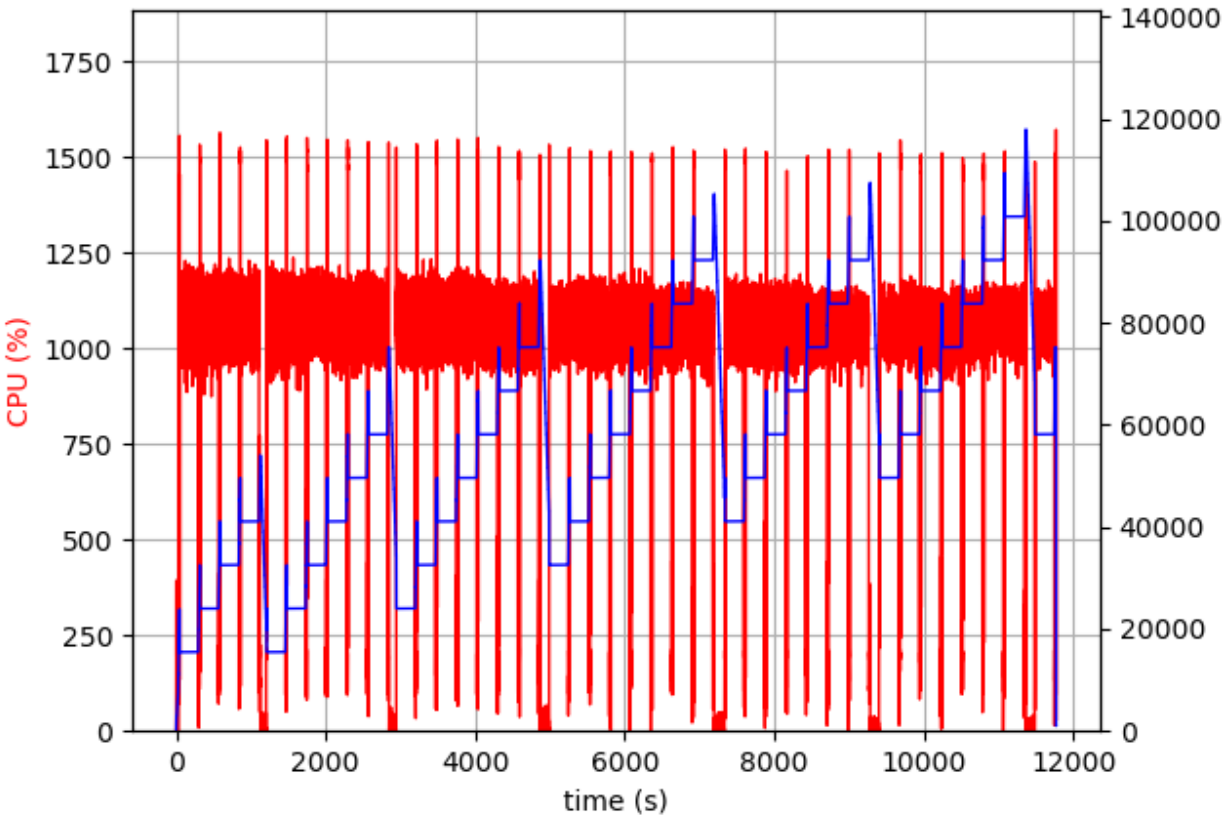




Running the distributed version of the code is slower than question 2, even with only a single node. My guess is that this is due to some additional overheads for the distributed classes from PyTorch. The multi node versions are slower still, this is due to the added cost of synchronizing weight updates between the nodes. To ensure a fair comparison, we should use a smaller batch size when there are more nodes, this is since the weight updates are aggregated between the nodes. If we have two nodes each with a bath size of 32, the effective batch size for each update is 64. The speedup we get from reducing the batch size is not as large as the added cost of synchronization. Also, with multiple nodes, we are limited by the slowest node.

Question 7:

11723.223086357117 seconds to train



Two worker processes are slower than a single process, this is again due to the cost of synchronizing the two processes. Also, both processes must share the resources of the node.

Competition:

For the competition, I modified the structure of the model to be a CNN with fewer parameters. I wrote my own training code in boilerplate.py which logs to the console differently than the starter code. I train for 14500 iterations with a batch size of 16. Training takes 86 seconds, which is slightly faster than training the same model on a single machine.

```
(pytorch_env) drakes@node0:~/eecs598/Assignment$ python PeerToPeer.py -n 3 -np 1 -a 10.10.1.1 -p 9955 --batch_size 16 -nr 0 -i 14500
Namespace(address='10.10.1.1', batch_size=16, data_dir='../../data/', do_chkpt=False, iterations=14500, local_rank=0, nodes=3, num_proc=1, port='9955', world_size=3)
torch.Size([6, 3, 5, 5])
torch.Size([6])
torch.Size([16, 6, 5, 5])
torch.Size([16])
torch.Size([64, 784])
torch.Size([64])
torch.Size([10, 64])
torch.Size([10])
Using downloaded and verified file: ../../data/extra_32x32.mat
Using downloaded and verified file: ../../data/train_32x32.mat
Using downloaded and verified file: ../../data/test_32x32.mat

-------+---------+-----------------+---------------------+---------------+------------+
| Epoch |    Rank | Loss/Validation | Accuracy/Validation | Learning Rate | Wall Time |
+=======+=========+=================+=====================+===============+============+
|     1 |       0 |          0.3633 |              0.9003 |      1.00E-02 |  00:01:15 |
+-------+---------+-----------------+---------------------+---------------+------------+
85.83595323562622  seconds to train

(pytorch_env) drakes@node1:~/eecs598/Assignment$ python PeerToPeer.py -n 3 -np 1 -a 10.10.1.1 -p 9955 --batch_size 16 -nr 1 -i 14500
Namespace(address='10.10.1.1', batch_size=16, data_dir='../../data/', do_chkpt=False, iterations=14500, local_rank=1, nodes=3, num_proc=1, port='9955', world_size=3)
torch.Size([6, 3, 5, 5])
torch.Size([6])
torch.Size([16, 6, 5, 5])
torch.Size([16])
torch.Size([64, 784])
torch.Size([64])
torch.Size([10, 64])
torch.Size([10])
Using downloaded and verified file: ../../data/extra_32x32.mat
Using downloaded and verified file: ../../data/train_32x32.mat
Using downloaded and verified file: ../../data/test_32x32.mat
+-------+---------+-----------------+---------------------+---------------+------------+
| Epoch |    Rank | Loss/Validation | Accuracy/Validation | Learning Rate | Wall Time |
+=======+=========+=================+=====================+===============+============+
|     1 |       1 |          0.3633 |              0.9003 |      1.00E-02 |  00:01:15 |
+-------+---------+-----------------+---------------------+---------------+------------+
85.7846212387085  seconds to train

(pytorch_env) drakes@node2:~/eecs598/Assignment$ python PeerToPeer.py -n 3 -np 1 -a 10.10.1.1 -p 9955 --batch_size 16 -nr 2 -i 14500
Namespace(address='10.10.1.1', batch_size=16, data_dir='../../data/', do_chkpt=False, iterations=14500, local_rank=2, nodes=3, num_proc=1, port='9955', world_size=3)
torch.Size([6, 3, 5, 5])
torch.Size([6])
torch.Size([16, 6, 5, 5])
torch.Size([16])
torch.Size([64, 784])
torch.Size([64])
torch.Size([10, 64])
torch.Size([10])
Using downloaded and verified file: ../../data/extra_32x32.mat
Using downloaded and verified file: ../../data/train_32x32.mat
Using downloaded and verified file: ../../data/test_32x32.mat

-------+---------+-----------------+---------------------+---------------+------------+
| Epoch |    Rank | Loss/Validation | Accuracy/Validation | Learning Rate | Wall Time |
+=======+=========+=================+=====================+===============+============+
|     1 |       2 |          0.3633 |              0.9003 |      1.00E-02 |  00:01:15 |
+-------+---------+-----------------+---------------------+---------------+------------+
85.8126814365387  seconds to train
```

To run

node 1 : python PeerToPeer.py -n 3 -np 1 -a 10.10.1.1 -p 9955 --batch_size 16 -i 14500 -nr 0
node 2 : python PeerToPeer.py -n 3 -np 1 -a 10.10.1.1 -p 9955 --batch_size 16 -i 14500 -nr 1
node 3 : python PeerToPeer.py -n 3 -np 1 -a 10.10.1.1 -p 9955 --batch_size 16 -i 14500 -nr 2