CS 744 Assignment 2 Report

Zifan Liu, Yufei Wang, Jiefeng Chen

Part 1 Logistic Regression

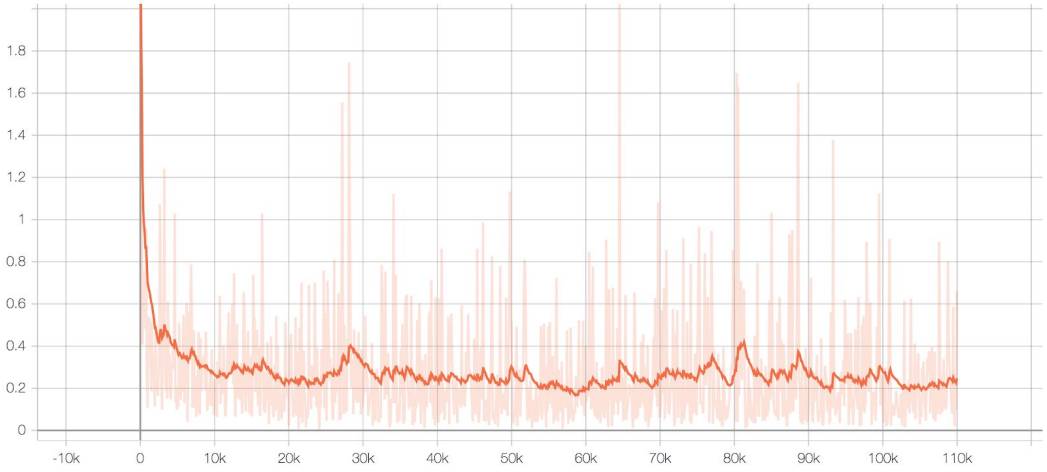
Task 2

In this task, we have implemented the logistic regression in synchronous and asynchronous modes. For each mode, we have recorded the loss and global steps per second using TensorBoard and monitored the usage of CPU/Memory/Network using dstat. The results are as follows (we run the experiment with 3 nodes, 20 epochs and batch size 10):

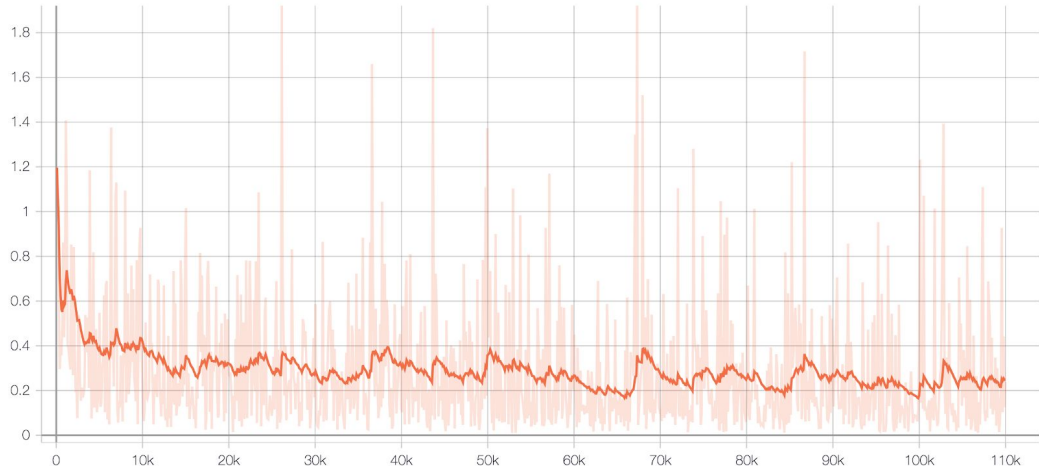
Loss with respect to the number of training steps

Synchronous mode:

(The light orange line represents the true value, and the dark orange line represents smoothed value, the same for all the figures from TensorBoard)

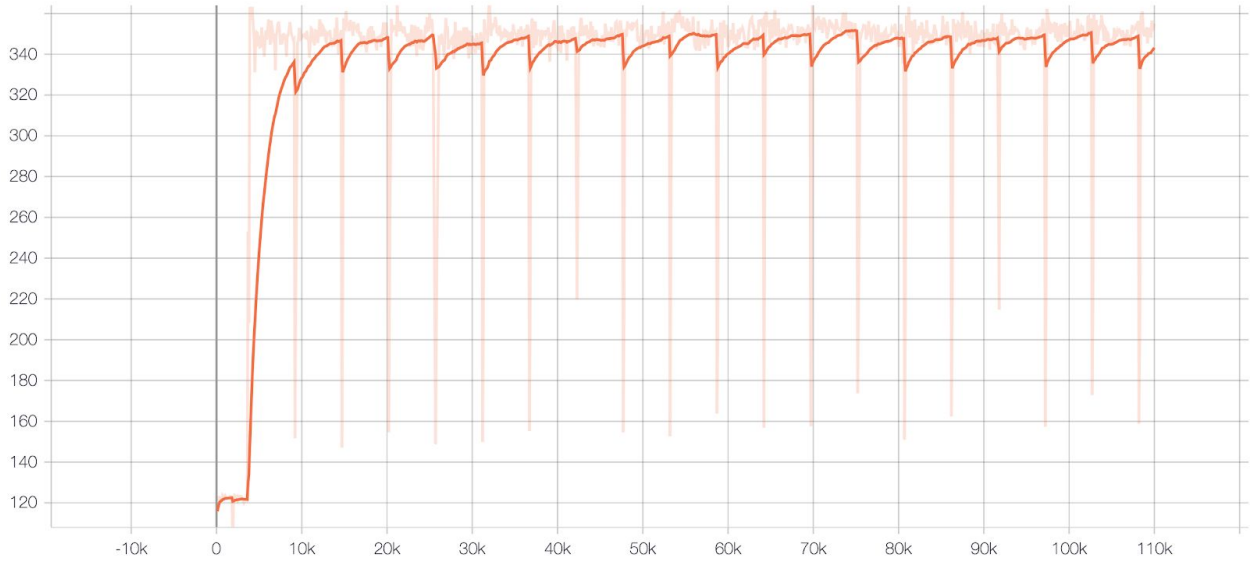


Asynchronous mode:

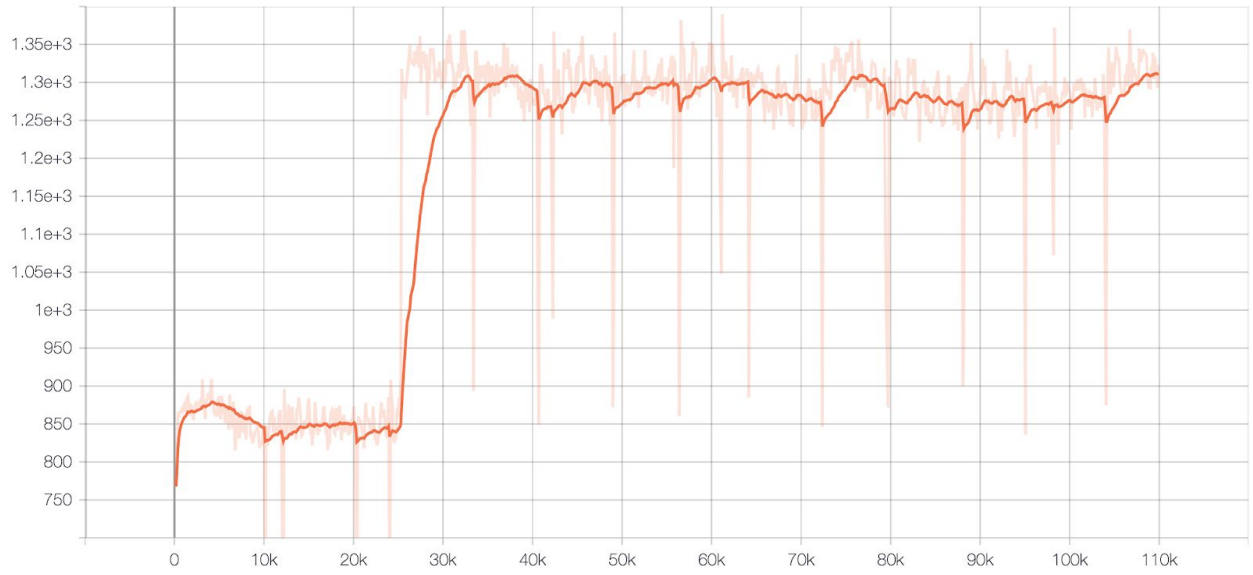


From the results above, we could see that the convergence rate in the synchronous mode is a little bit faster than the asynchronous mode (with respect to the number of training steps), and the loss in the asynchronous mode is noisier. The reason for this is that in the asynchronous mode, the update may not be based on the latest parameters.

Global steps/second with respect to the number of training steps
Synchronous mode:



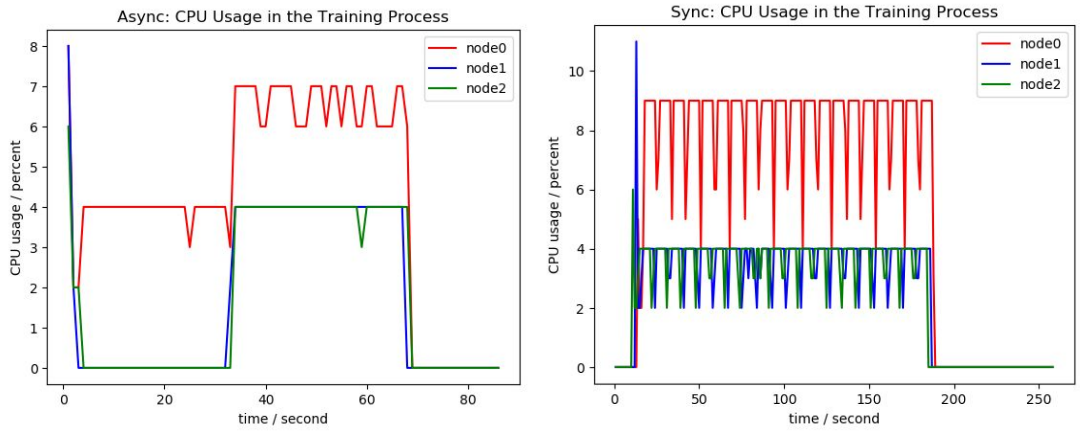
Asynchronous mode:



From the results above, we could see that the speed of the asynchronous mode (global steps per second) is 3X faster than the synchronous mode. We could also notice that at the beginning of asynchronous training, the speed is about 850 global steps per second, and after about 30

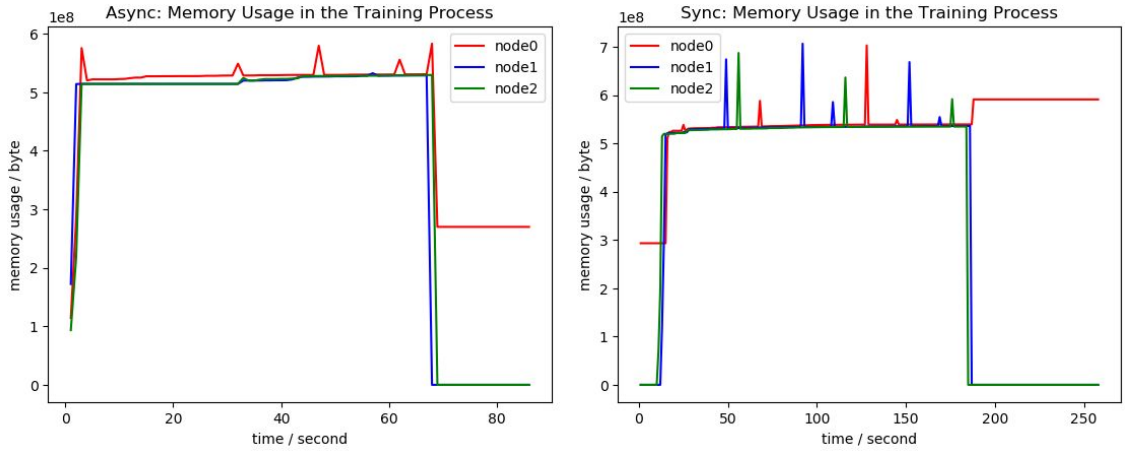
seconds it finally reaches about 1300 global steps per second. We tried to monitor what has happened in the training process, and we found that some worker machine may not start at the beginning of the training process and will reconnect after 30 seconds, which is a feature of MonitoredTrainingSession (link).

CPU usage



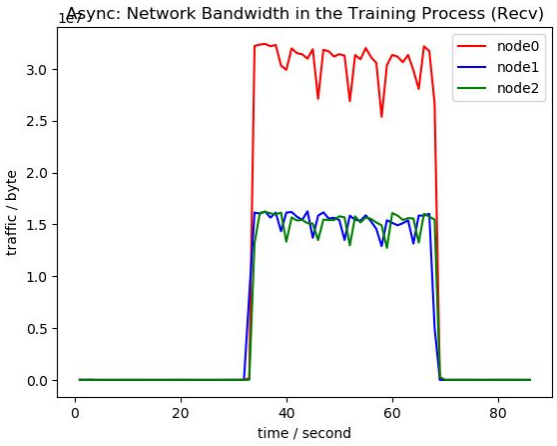
As expected, the CPU usage of node0 is higher than the other two because node0 is the chief node and the parameter server is in node0. In the synchronous mode, the CPU usage of node0 is higher than that in the asynchronous mode because of the cost of synchronization. We have also observed a periodic pattern of CPU usage in the synchronous mode - the CPU usage will drop at the end of each epoch.

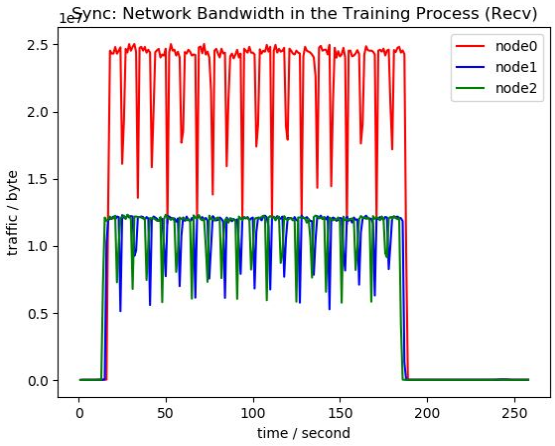
Memory usage

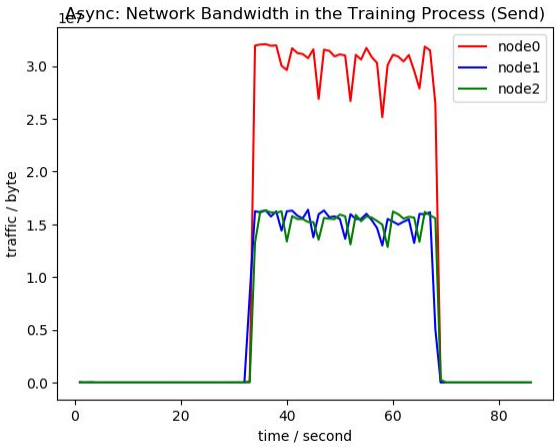


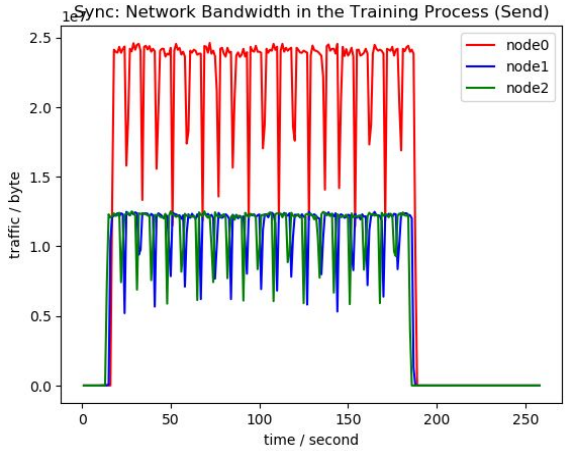
The memory usage is stable, and there is no notable difference between the two modes.

Network Usage









The network traffic of node0 is approximately the sum of node1 and node2, because node0 is responsible for sending parameters and receiving updates. In the synchronous mode, the network usage has a periodic pattern, which is really similar to the CPU usage.

From the statistics above, the bottleneck of the system is network. Besides, in synchronous mode, the overhead of synchronization is also not negligible.

Task 3

In this task, we also use dstat to record the network traffic, CPU usage and memory usage in the training process with different batch size. For simplicity, we compute the average of the chief node's CPU usage, memory usage, and network traffic, which is illustrated below:

Asynchronous:

Batch Size	5	10	20	30	40
CPU (%)	6.93	6.52	6.76	6.89	6.67
Send (MB/s)	31.944	31.310	30.352	28.383	28.732
Recv (MB/s)	31.354	31.091	29.644	28.183	28.493
Memory (MB)	509.422	498.379	505.715	503.019	502.894
Accuracy (%)	92.24	92.41	92.40	92.28	92.12

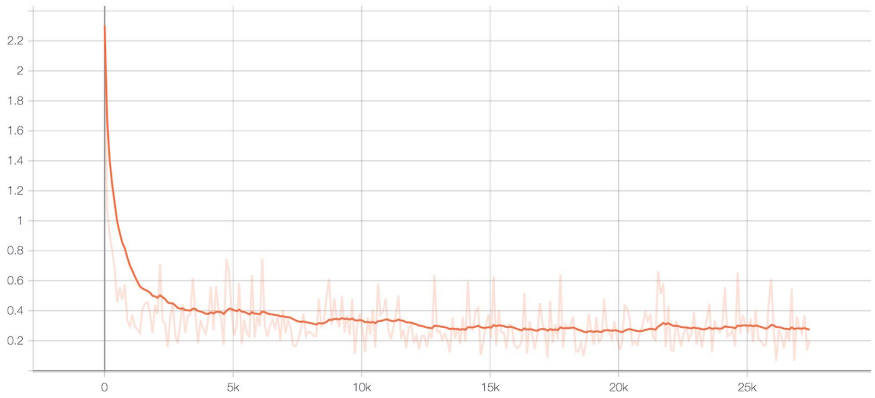
Synchronous:

Batch Size	5	10	20	30	40
CPU (%)	8.81	8.75	8.42	7.24	7.03
Send (MB/s)	25.468	24.901	24.532	21.487	20.668
Recv (MB/s)	25.891	25.331	24.049	20.234	20.318
Memory (MB)	541.327	540.553	539.308	533.127	535.894
Accuracy (%)	92.39	92.54	92.31	92.24	92.23

We could find that in asynchronous/synchronous mode, the CPU usage and memory usage remains stable for different batch sizes. Besides, the accuracy of models trained with different batch sizes still seems to be the same.

The thing which is different for different batch sizes is network traffic. When batch size becomes larger, the network traffic seems to be slightly lower. We think this is as expected because when the batch size becomes larger, there will be fewer updates transferring through the network. What's more, when the batch size becomes larger, it will cost less time to finish the training process.

As to the loss, it is more stable with larger batch size. The figure below shows the loss change when the batch size is 40.

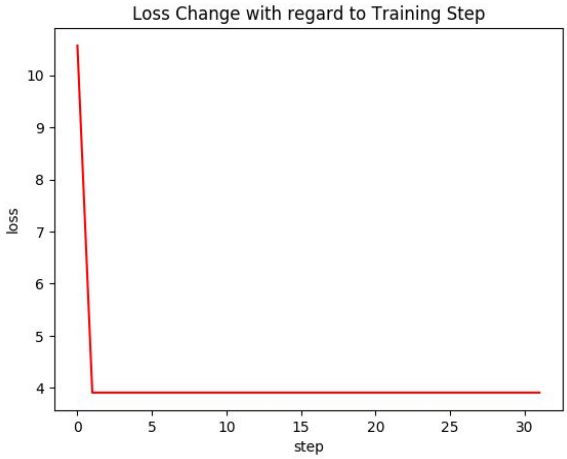


Part 2 AlexNet

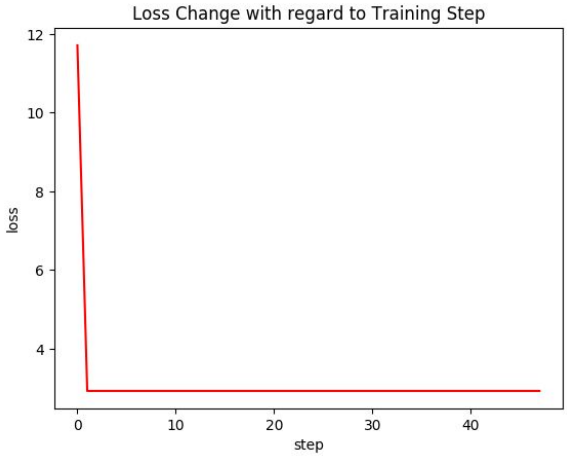
Task 1

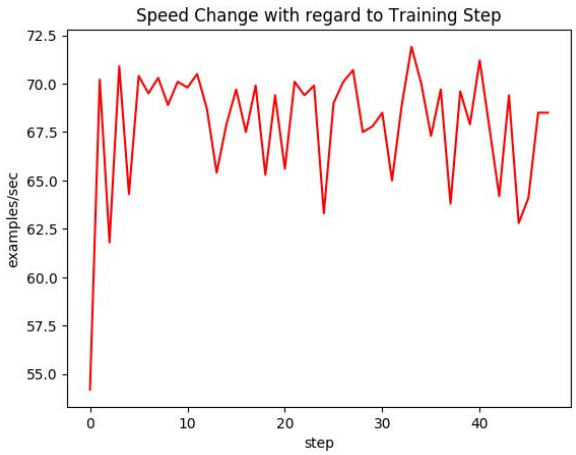
Performance

- Run AlexNet using 3 machines



- Run AlexNet using 2 machines





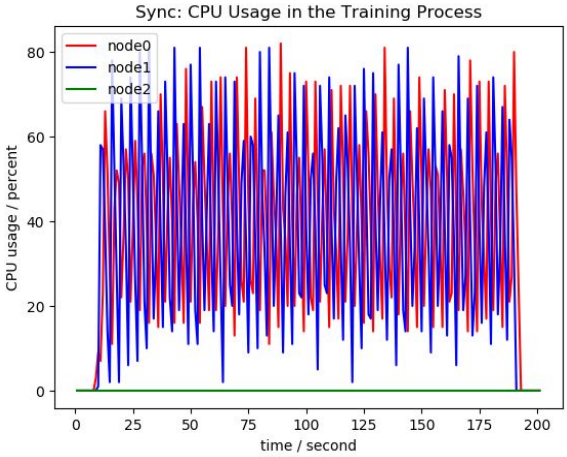
- Comparison

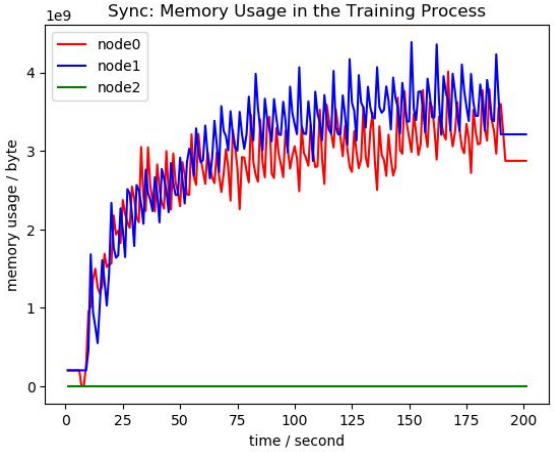
From the results, we can see that the speed of using 3 machines is approximately 1.5 times as fast as that of using 2 machines. And the loss changes of both cases are similar.

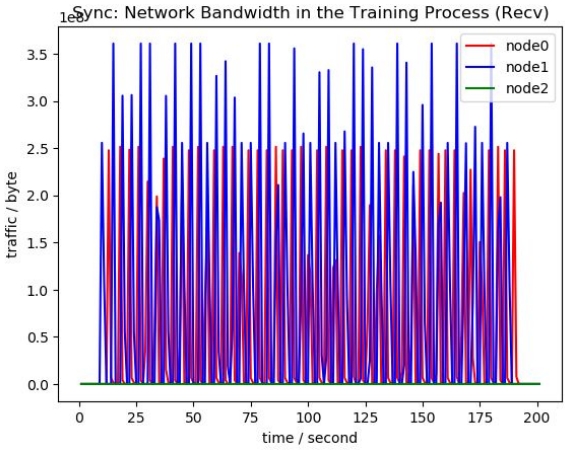
Task 2

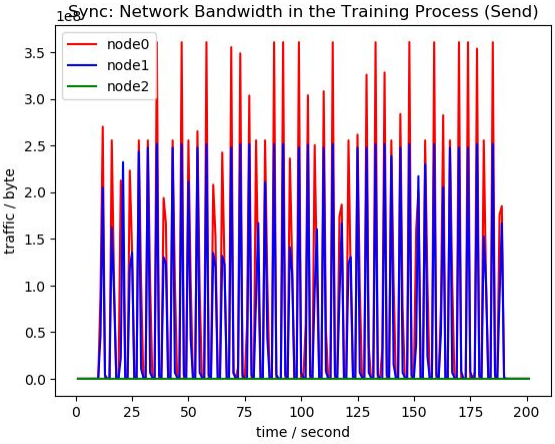
CPU/Memory/Network usage

- Run the AlexNet using two machines

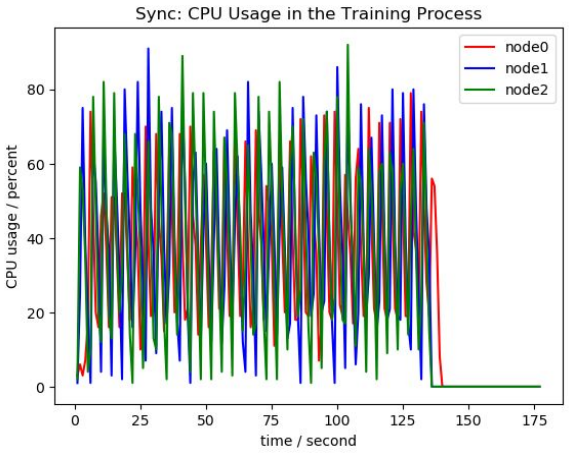


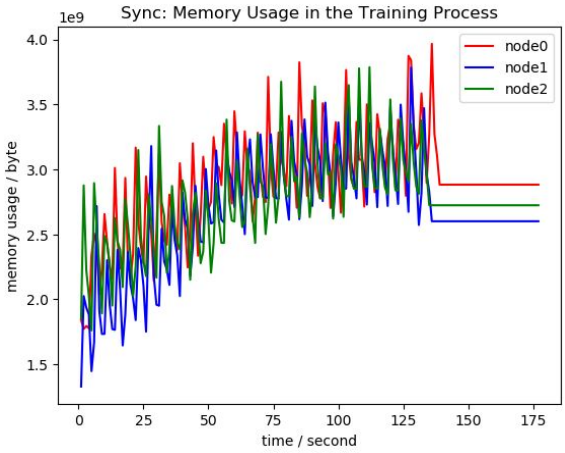


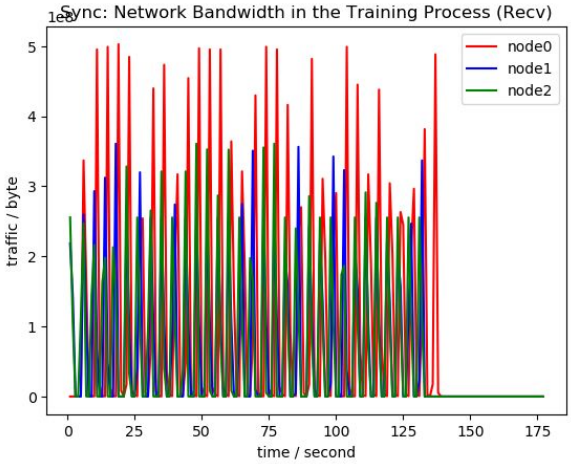


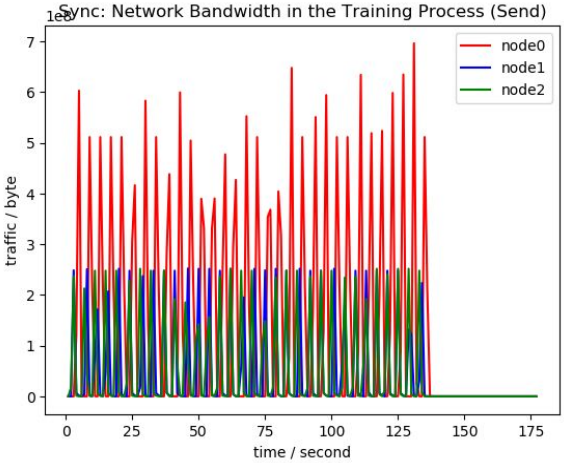


- Run the AlexNet using three machines









- Comparison

(1) CPU usage: the performance of using two machines is similar to the performance of using three machines.

(2) Memory usage: the performance of using two machines is similar to the performance of using three machines.

(3) Network usage: the network traffic in node0 of using three machines is heavier than that of using two machines. This is expected, as in three machines case, node0 needs to send/receive messages to/from two nodes while in two machines case, node 0 only needs to send/receive messages to/from one node. The network traffic in node1 and node2 of using three machines is similar to that of using two machines.