CS744-Assignment2

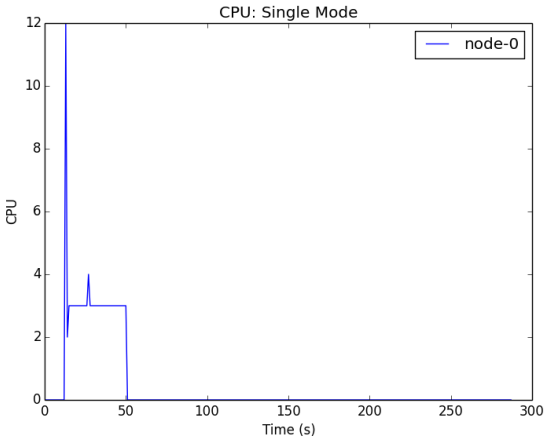
Pan WU, Yiwu Zhong, Zihang Meng

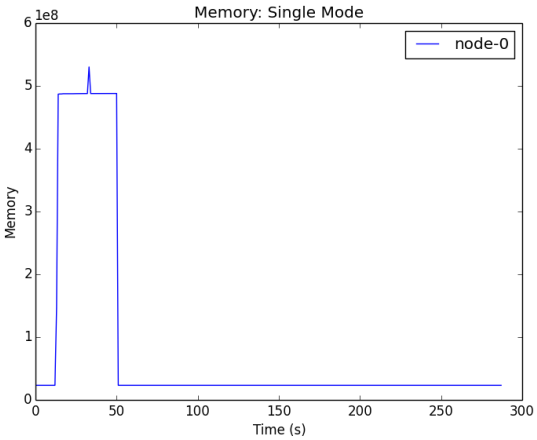
April 19, 2019

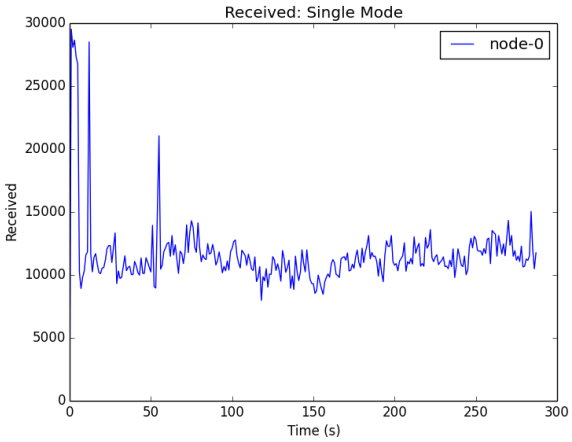
1 Part1 Logistic regression

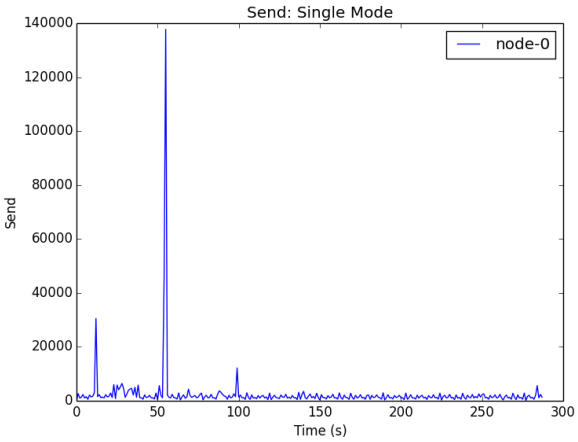
1.1 Task1(Single Mode)

Performance:
batch size = 32
epoch = 20
Training time = 37.2255s
Accuracy = 0.9203
CPU / Memory / Network usage:









1.2 Task2(Sync SGD and Async SGD)

Async:

Performance:

workers number = 3

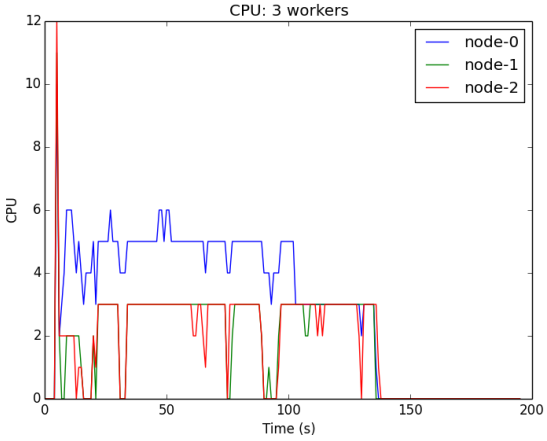
batch size = 32

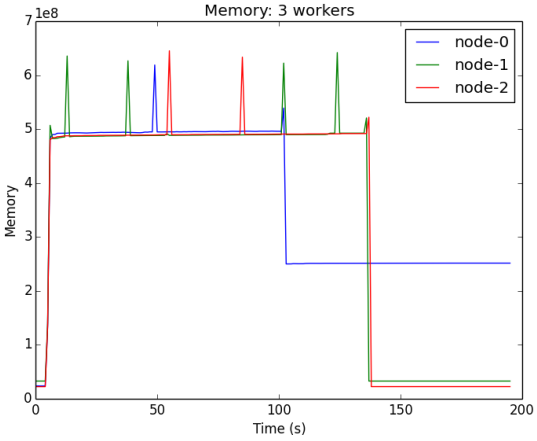
epoch = 20

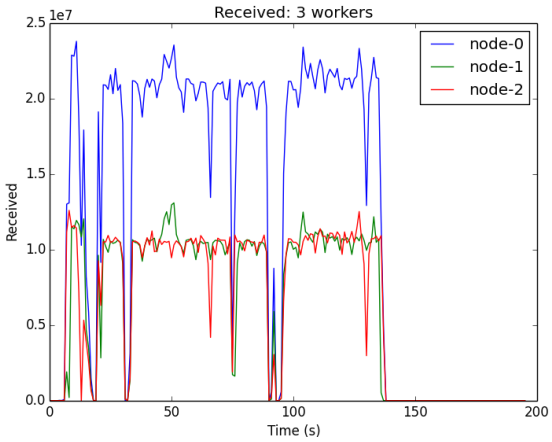
Training time = 95.1249s

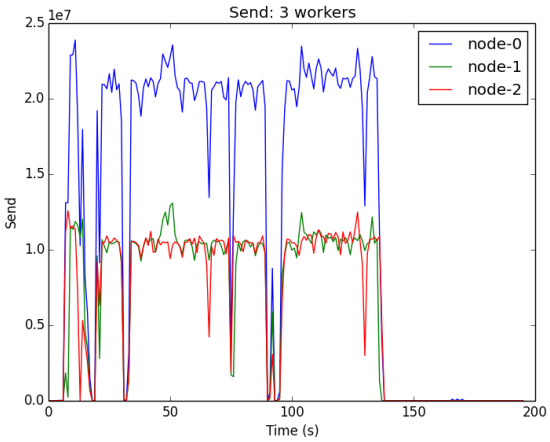
Accuracy = 0.9235

CPU / Memory / Network usage:

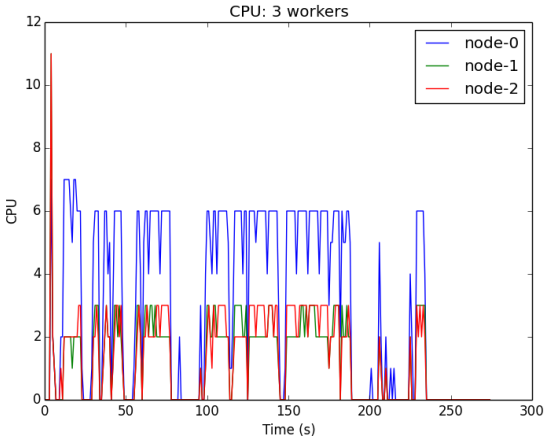


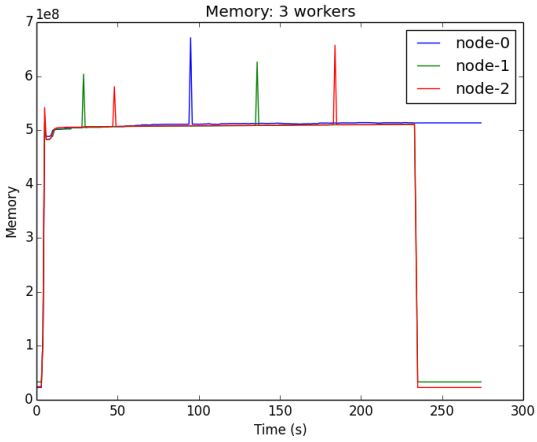


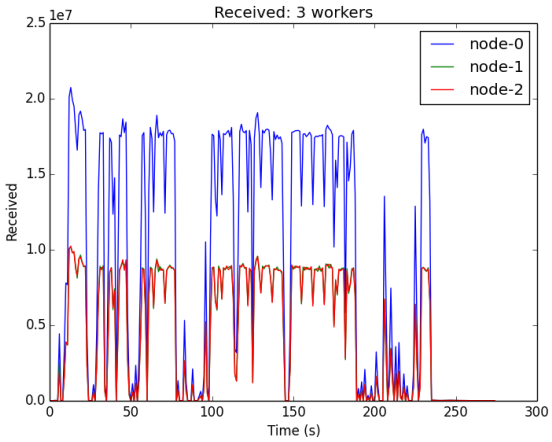


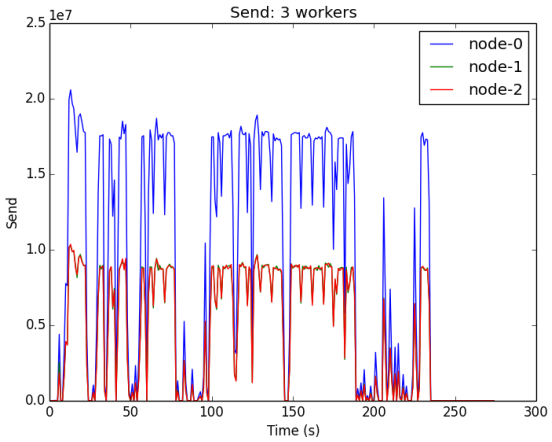


Sync:
Performance:
workers number = 3
batch size = 32
epoch = 20
Training time = 230.2578s
Accuracy = 0.9199
CPU / Memory / Network usage:









1. Sync mode is slower than Async mode with same epoch number and same batch size. This is because in Sync mode, finished worker need to wait other workers in the same step and wait the aggregation, after that, then the workers can start next step. So Sync mode is expected to be slower. Our experiment proved that.

2. In terms of test error, the Sync mode and Async mode are similar (91.99% and 92.35% respectively), which makes sense because they are using the same dataset and the number of epochs are the same.

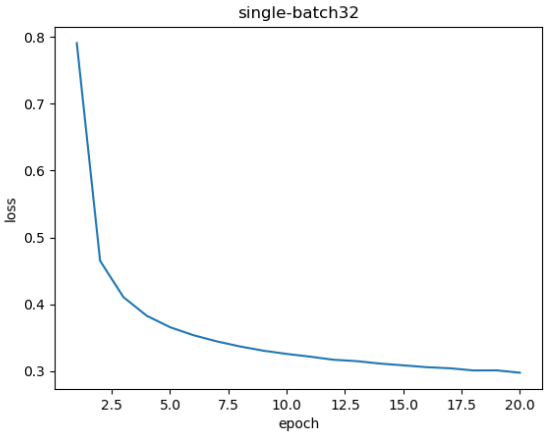
3. The CPU and network usage of node-0 is larger than that of node-1 and node-2, and the network of node-0 is almost equal to the sum of node-1 and node-2. That's because node-0 is the parameter server, and it send and receive data from all workers.

4. The fluctuation of the CPU and network usage of Sync mode is bigger and more frequent than that of the Async mode. That make sense, because in Sync mode, workers need to wait, and in these periods, the CPU and network stop working, so the fluctuation will be big and more frequent for Sync mode.

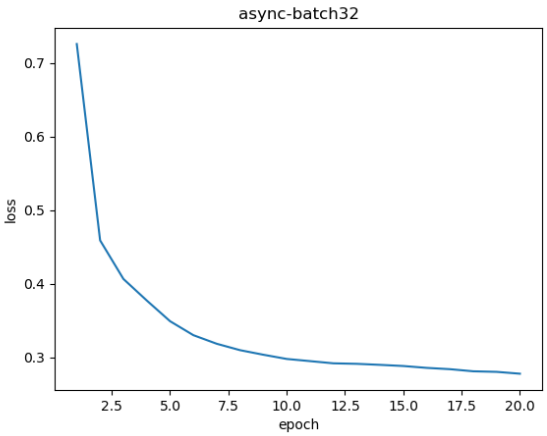
5. As we can see, we don't have obvious bottleneck since network bandwidth has never been fully utilized and memory is also only partly used. The only thing left is cpu frequency. If we increase the cpu frequency, then obviously the execution will be faster.

1.3 Task3(different batch size)

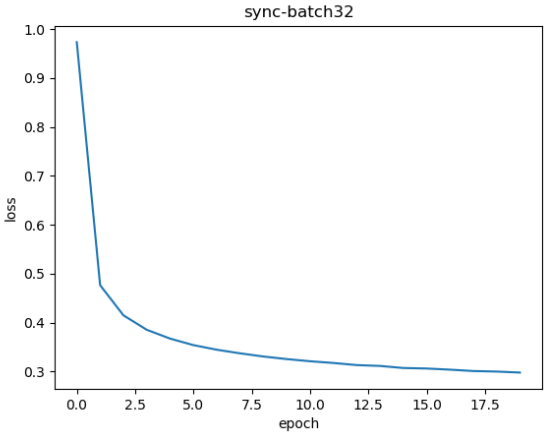
batch size = 32:
Single mode
training time = 37.2255s



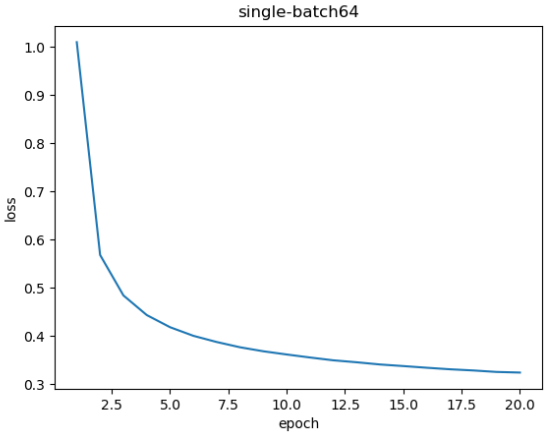
Async
training time = 95.1249s



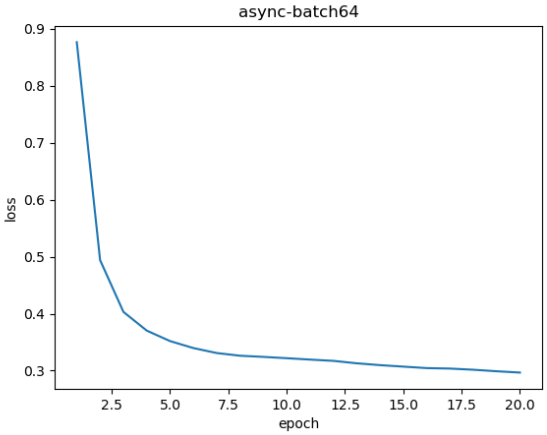
Sync
training time = 230.2578s



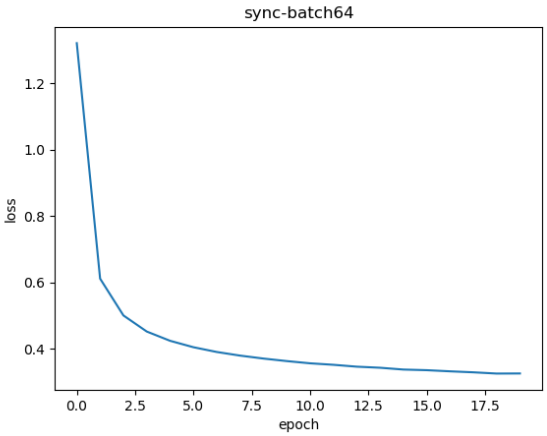
batch size = 64:
single mode
training time = 19.0367s



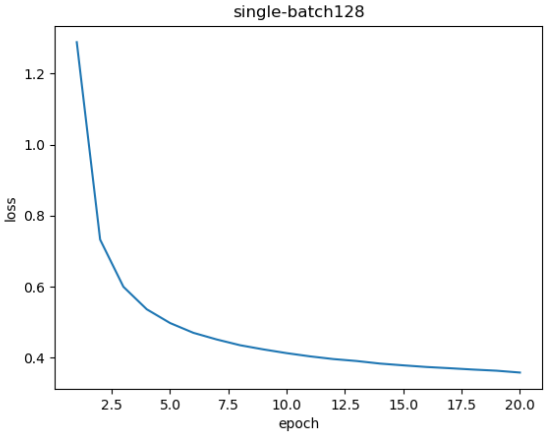
Async
training time = 53.6177s



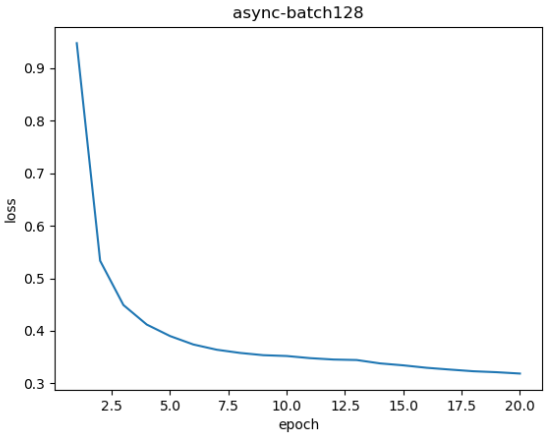
Sync
training time = 157.6993s



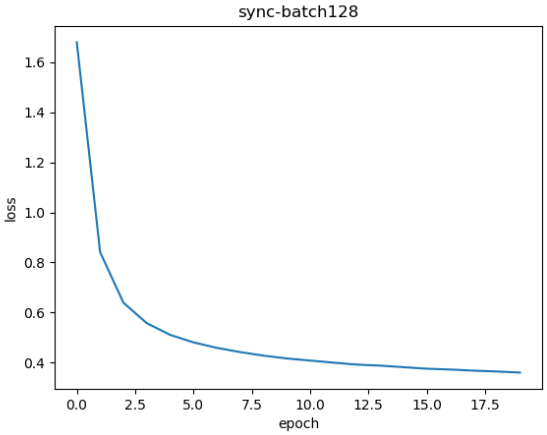
batch size = 128:
single mode
training time = 17.7009s



Async
training time = 32.7711s



Sync
training time = 110.6342s



As we can see, the Async mode is always faster than Sync mode, due to the same reason in task2. And the speed increases as the batch size increase because larger batch size will utilize CPU computing resource more sufficiently. In terms of training loss, batch size 32 is a little better than 64 and 128.

2 Part2 AlexNet

2.1 Task1: 3 workers

The loss only decreases for **the 1st iteration** due to the "fake data" used in experiment. The training processed **89.5** examples per second.

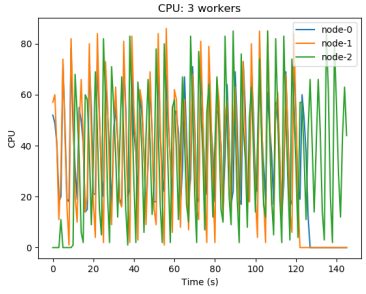


Figure 1: CPU: 3-worker

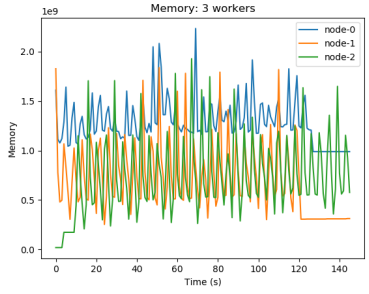


Figure 2: Memory: 3-worker

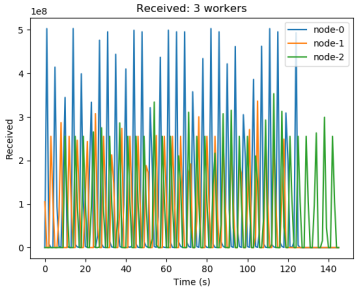


Figure 3: Received: 3-worker

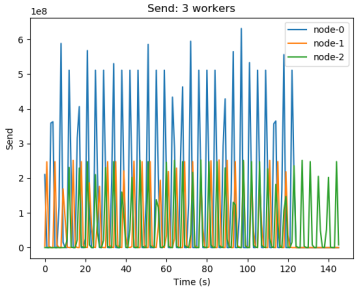


Figure 4: Send: 3-worker

Period: Since we use synchronous mode in this part, the cycles of CPU computation, memory usage and I/O of each worker are almost the same and vibrate simultaneously.

Amplitude: The CPU usage among 3 workers are almost the same, which means each worker has almost same computation. The memory used in node-0 is always higher because it not only runs as a worker but also as the PS server applies all gradients which received from other workers to the model. The network traffic of node-0 is roughly the sum of the other 2 workers because node-0 send models and received updates from the other 2 workers.

According to the figures, there is no obvious bottleneck. If we use some better CPUs with higher frequency, then the number of examples processed per second.

2.2 Task2: 2 workers

The loss only decreases for **the 1st iteration** due to the "fake data" used in experiment. The training processed **65.2** examples per second.

Compared to 3-worker scenario, 2-worker process **roughly 2/3 examples per second**, since basically the more workers which are used in the job, the quicker the job will be done, when the total example number keeps the same.

For CPU usage, node-0 and node-1 are still almost the same as using 2 workers. For memory usage and network usage, the difference between node-0 (PS) and node-1 is smaller than that of using 3 workers. That's because the more workers we use, the more gradient updates will be received and applied to model by the PS.

Figure 5: CPU: 2-worker

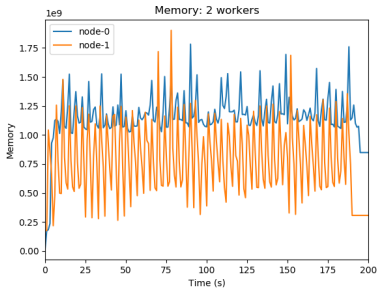


Figure 6: Memory: 2-worker

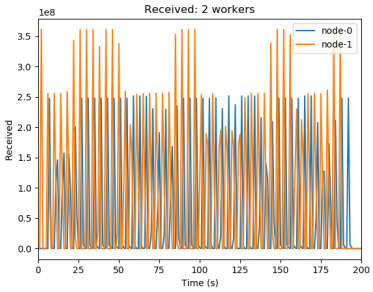


Figure 7: Received: 2-worker

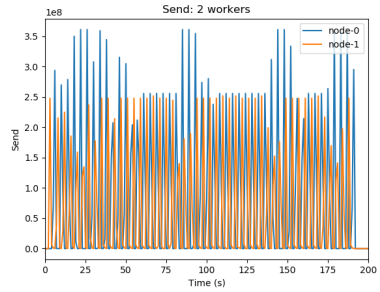


Figure 8: Send: 2-worker