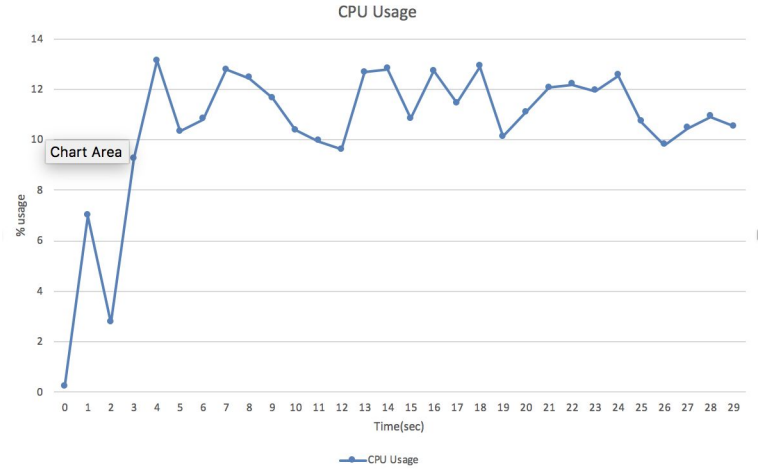
Assignment 2: TensorFlow

CS 744 : Big Data Systems
Group 3:
Yudhister Satija, Bidyut Hota, Venkatesh Somyajulu

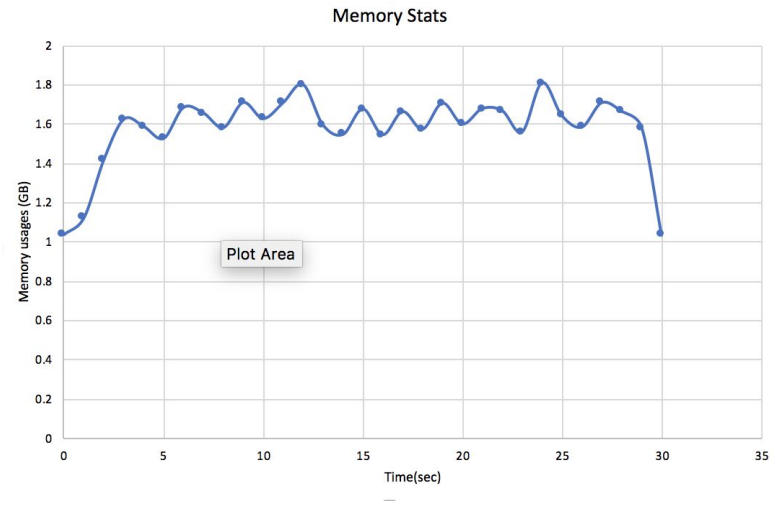
Part 1: Logistic regression

Task 1: LR on single node,

1. 100 epochs
    a. Time taken: 35s
    b. Accuracy on Test set: 0.870
    c. Loss on Test Set: 5.559
    d. CPU usage statistics



e. Memory usage



Task 2: Distributed LR, 100 epochs,4 nodes, data divided equally between nodes:

Mode	Time Taken (s)	Test Accuracy	Test Error
Synchronous	50	0.861	5.606
Asynchronous	18	0.871	5.227

We see that Synchronous mode waits for few seconds before the training loop begins. Because of this, the total runtime of the operation increases. Async mode does not have to wait, and each worker can begin training quickly.

Async mode convergence is expected to be slower, compared to Sync mode. Async mode receives stale model and hence the training could take longer to converge. But our measurements on the given dataset do not show any measurable difference as the dataset is too small.

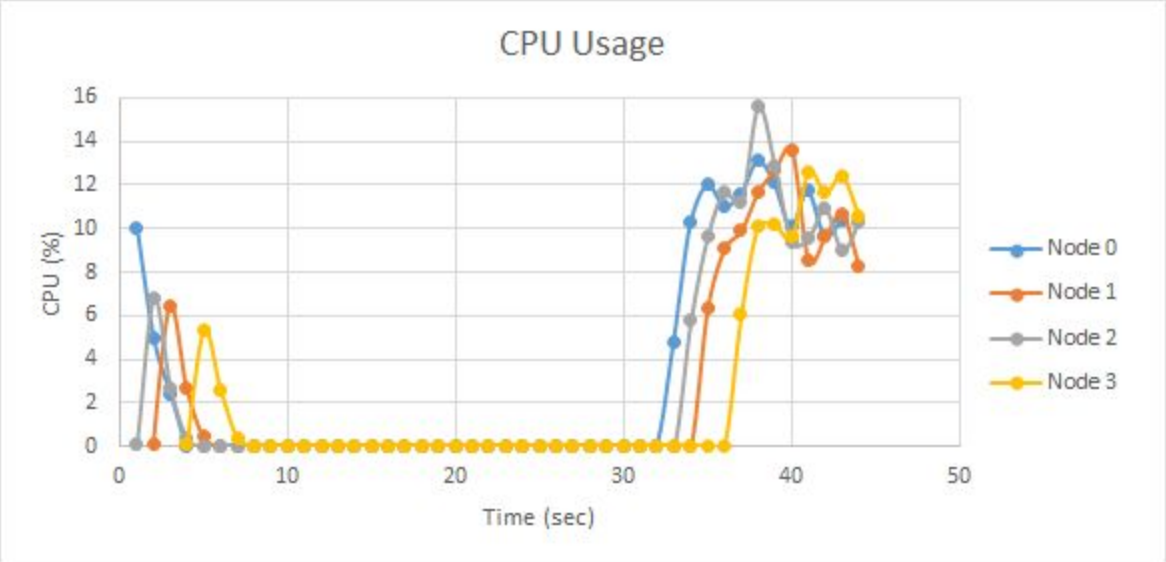
Accuracy and error plots over the epochs are provided in Task 4 section.

CPU / Memory / Network usage:

- Synchronous mode

  We see a 30 second wait period between server start on the nodes and actual computation beginning on them. This is due to TensorFlow nodes waiting for the same variables that are initialized by the session running on chief worker. This time limit is set by tensorflow when running in synchronous mode. In larger machine learning tasks, this is useful, but not so much on this small data set.

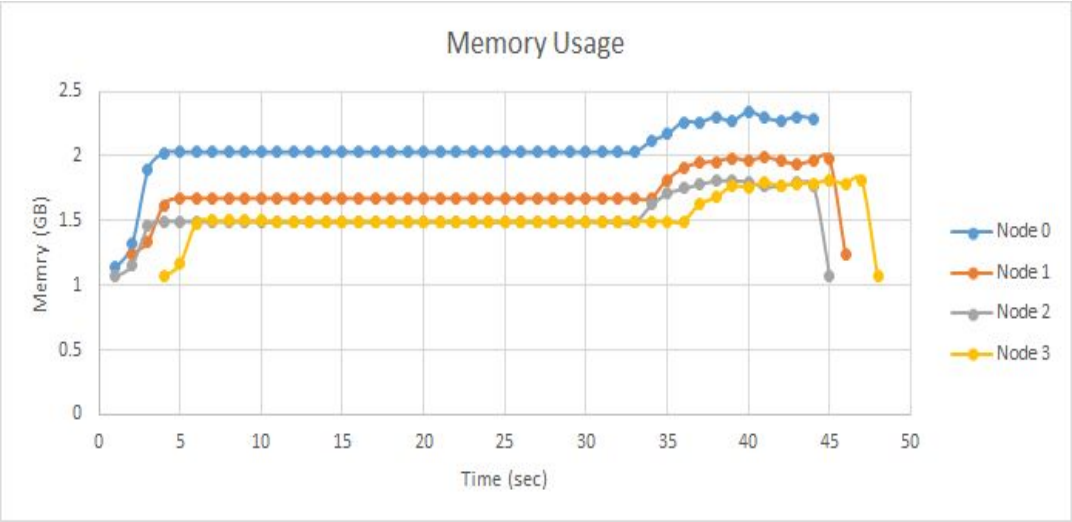
  - CPU usage



Initial usage of processor is high since the model is getting initialized. CPU usage goes to almost zero in the 30 second wait period. Once the nodes are synchronized, the CPU usage

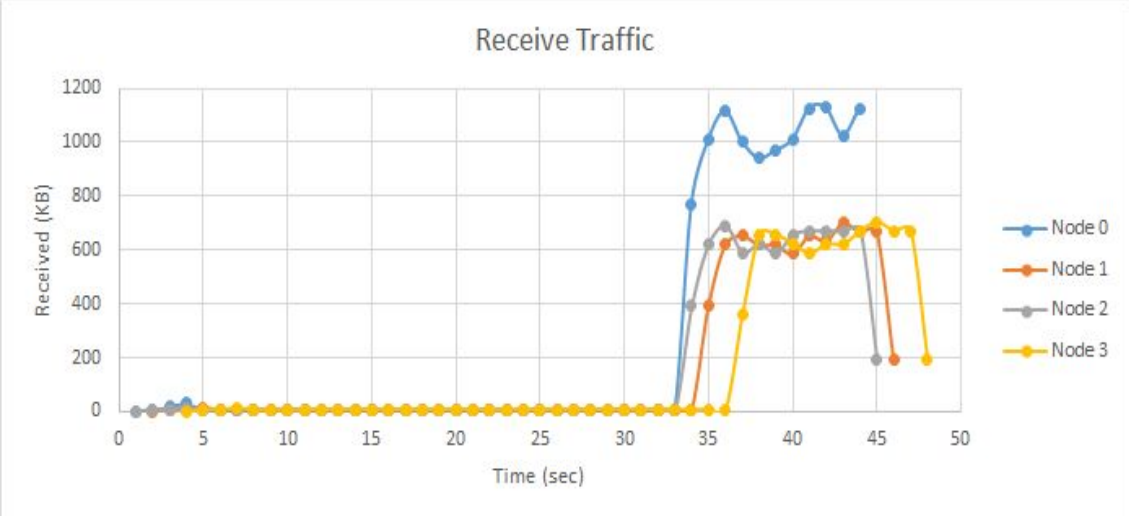
again witnessed an increase because the training starts. The order of the plots of the nodes is in the order of them getting the data and is slightly off by a couple of seconds.

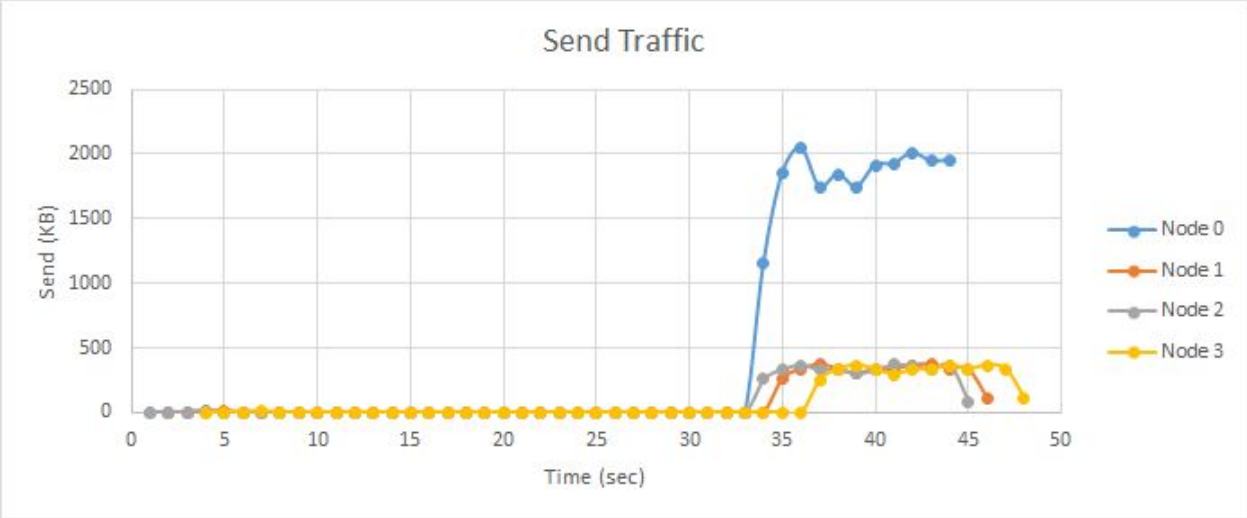
- Memory Usage



Memory usage is highest on Node0 since it runs the PS server as well as the worker node. The last part of the plots sees an increase in memory usage due while training.

- Network traffic

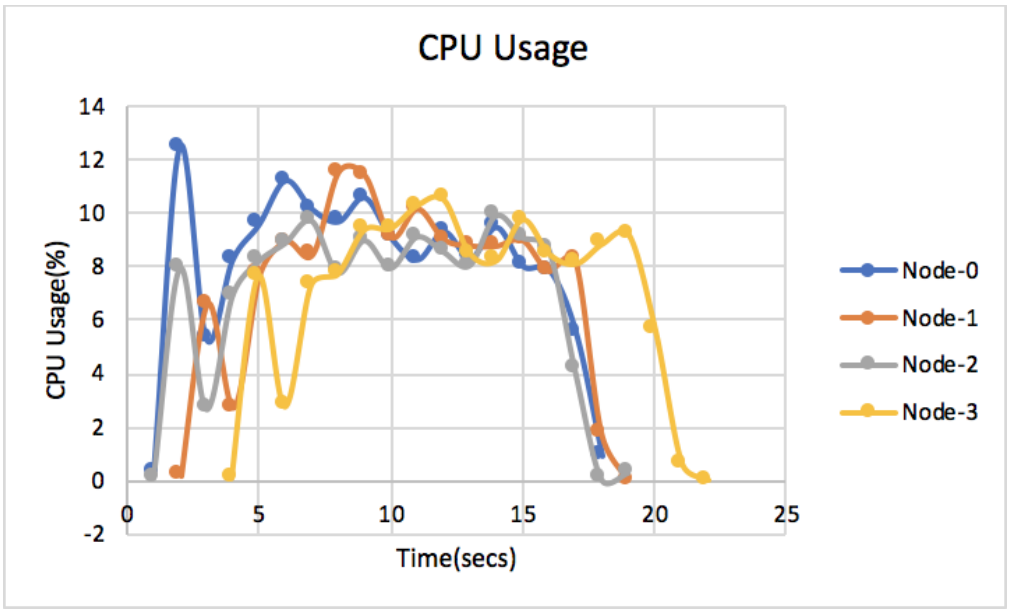




Just like previous plots, network traffic sees an initial wait period. Thereafter, we see that data sent out by node-0, which holds the parameter server and sends out the entire model, is more than thrice the other nodes, which send out only the model updates.

The Data received by node-0 is only slightly higher, because it receives the model updates from other 3 nodes. The other nodes only receive the entire model.

- Asynchronous mode
  - CPU usage



CPU usage at all 4 nodes is close, and we do not see any startup delays. All nodes start processing data as soon as they receive the python command to start.

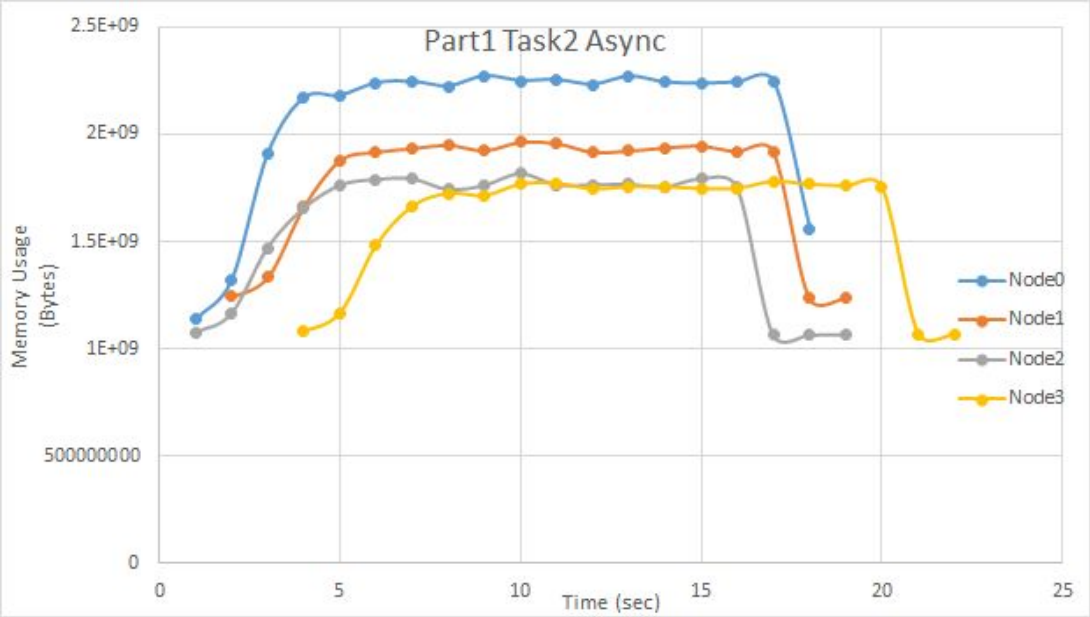
  - Network usage graphs collected via dsat:

The chief server running on Node-0 sends out the entire model to each of the other 3 nodes. The other nodes only send updates back to the model, from each worker. Hence, as expected, send traffic from node0 is more than 3 times the send traffic from other nodes.

The receive traffic values are closer. Node0 receives updates of the model from 3 nodes while others receive the the entire model from node0.

- Memory utilization

  The memory usage on node 0 is higher than the other due to the ps server also running apart from the worker process. The other three nodes have comparable memory usages. Once the training on the nodes ends, the memory usage starts declining to original levels.



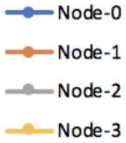
Task 3: Async mode, 4 nodes, Changing batch sizes

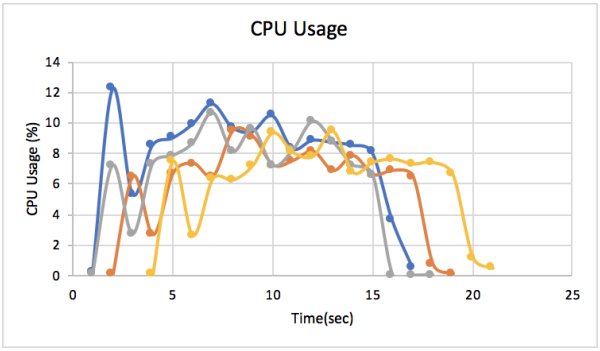
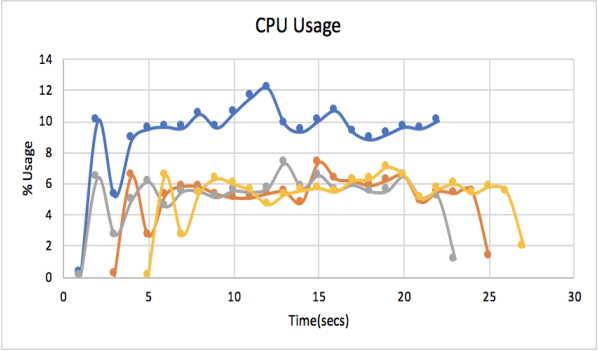
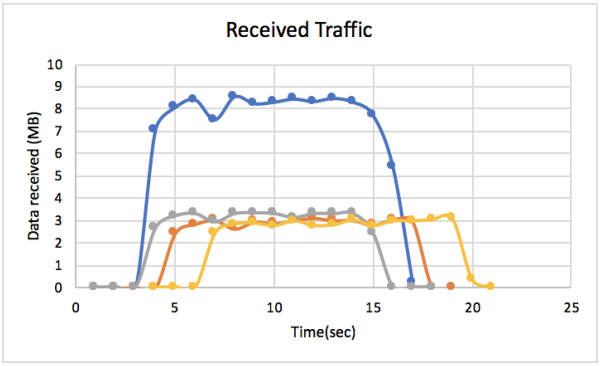
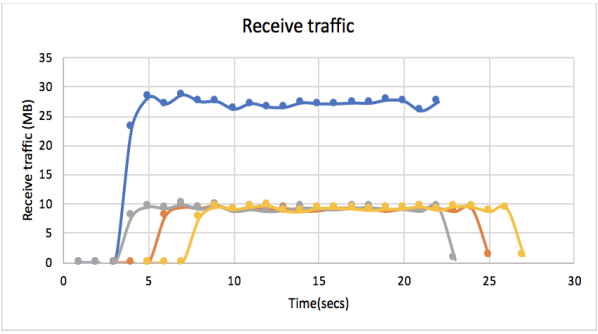
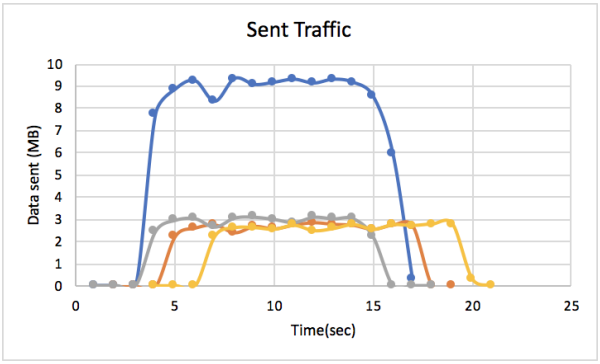
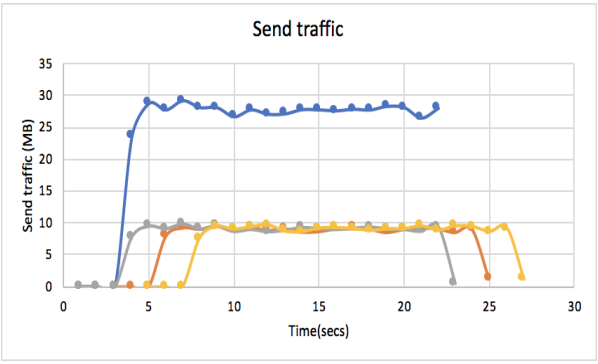
Num Batches	Batch Size	Time Taken (s)	Test Accuracy	Test Error
1	13750	18	0.871	5.227
10	1375	20	0.969	0.732
50	275	28	0.980	0.438

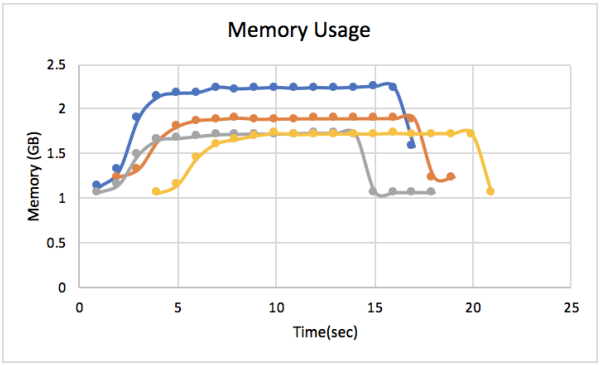
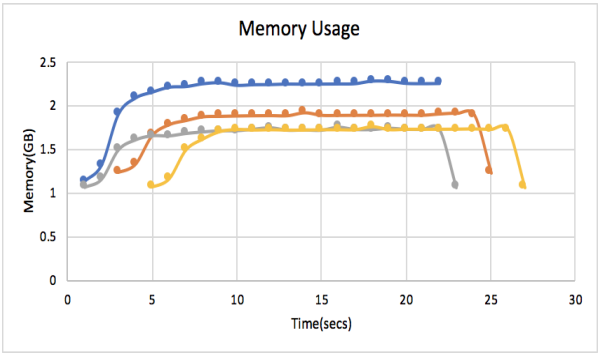
We see that with smaller batch size, the time taken increases as model needs to be updated more times per epoch. However, this also gives better accuracy and test error as the SGD algorithm is more suited to smaller batch sizes (using single data points in ideal case)

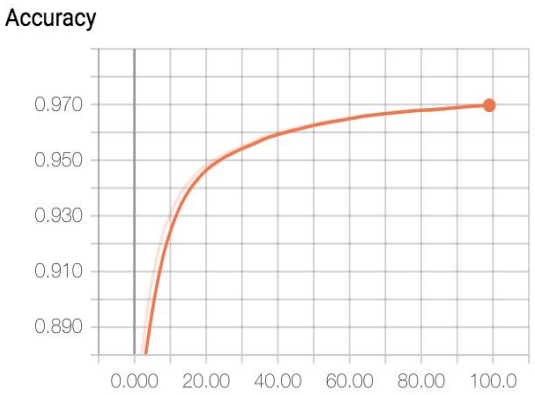
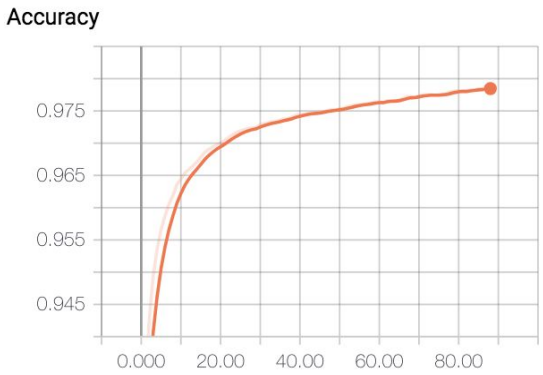
Comparison of CPU, Network and memory usage across different batch sizes:

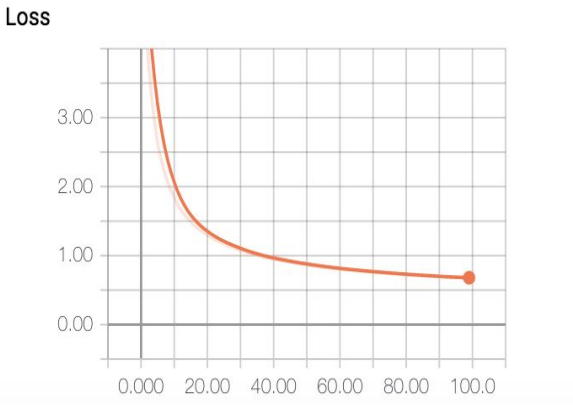
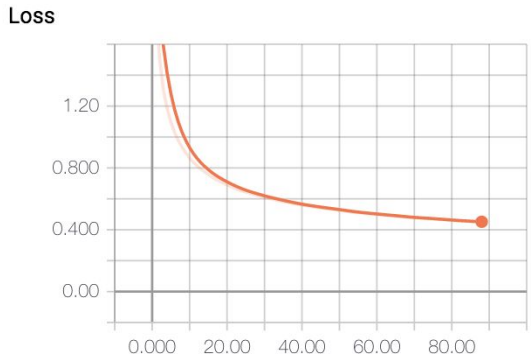
- Number of batches= 1: refer to task 2 above



Number of Batches = 10	Number of Batches = 50
Batch Size = 1375	Batch Size = 275
	 CPU usage increases with smaller batches, as more iterations are run
	 Network traffic increases as the number of times the PS is communicated increases (every call to optimize)
	 Same as above

<table>
<tr>
<td> 
</td>
<td> 

This graphs are similar as the model size does not change with batch size</td>
</tr>
<tr>
<td> 
</td>
<td> 

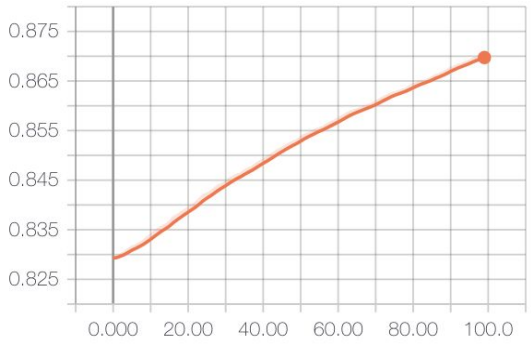
Explained above</td>
</tr>
<tr>
<td> 
</td>
<td> 

Explained above</td>
</tr>
</table>

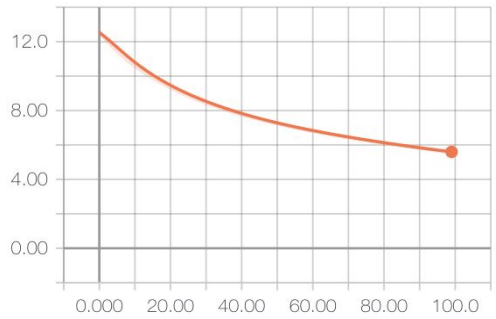
Task 4: Graphs from Tensor board

- Single node
    a. Accuracy on Test Set

Accuracy



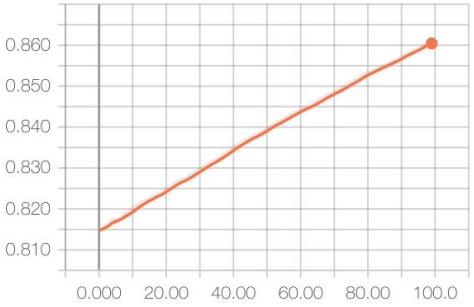
Loss



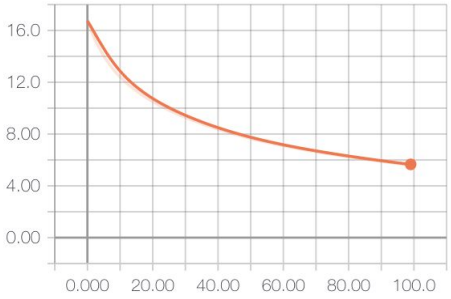
b. Loss on Test set

- Synchronous, 4 node
  a. Accaccuracy and Loss on Test Set

Accuracy

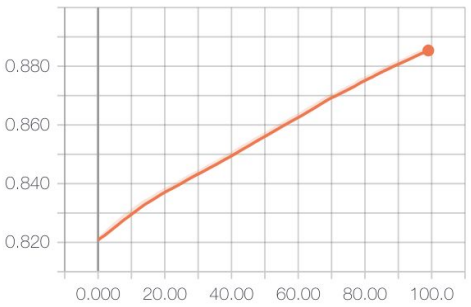


Loss

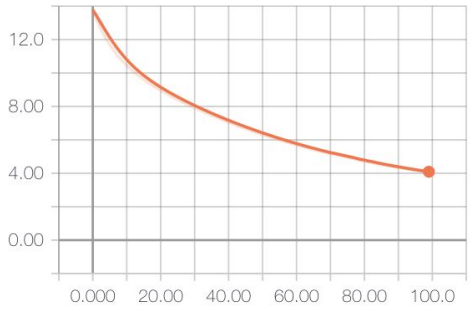


- Asynchronous, 4 node,
  a. Accuracy and Loss on Test Set

Accuracy



Loss



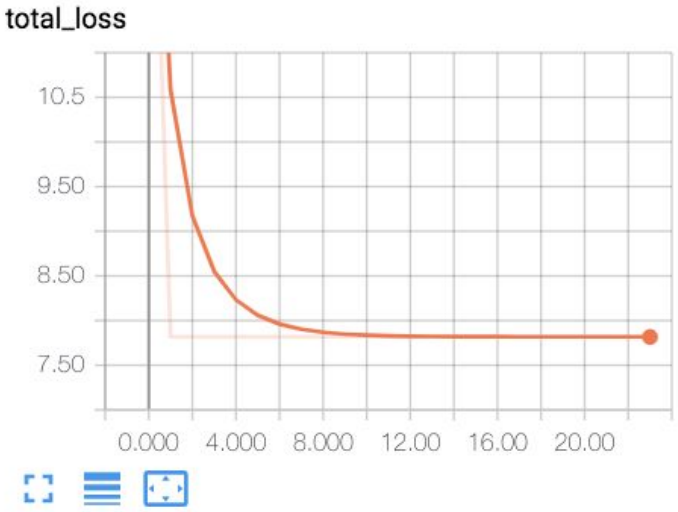
Part 2 AlexNet

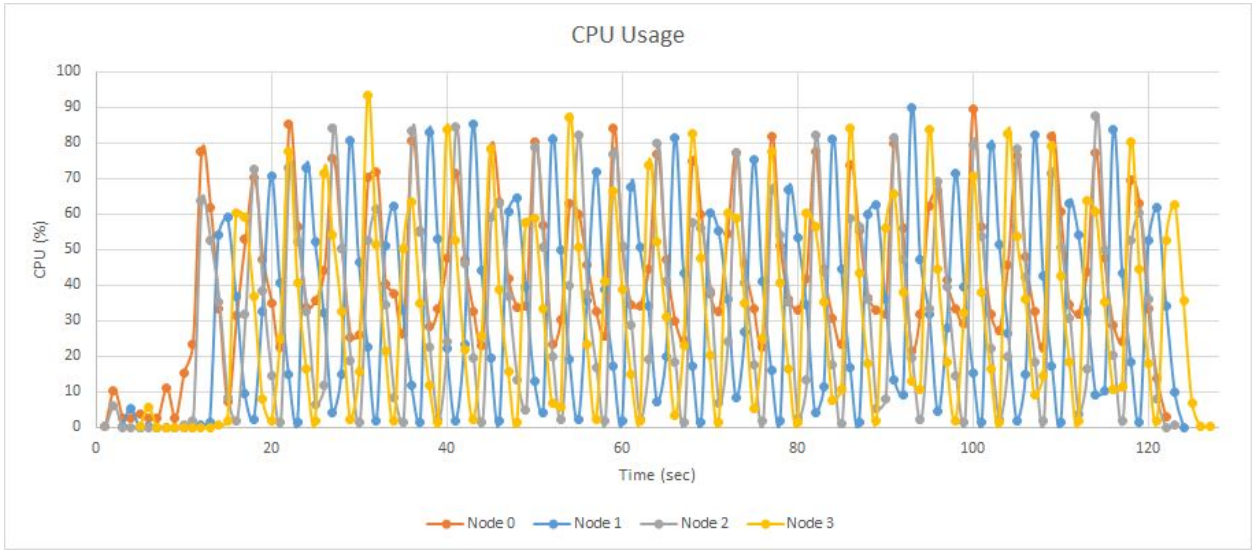
Task 1: 4 nodes

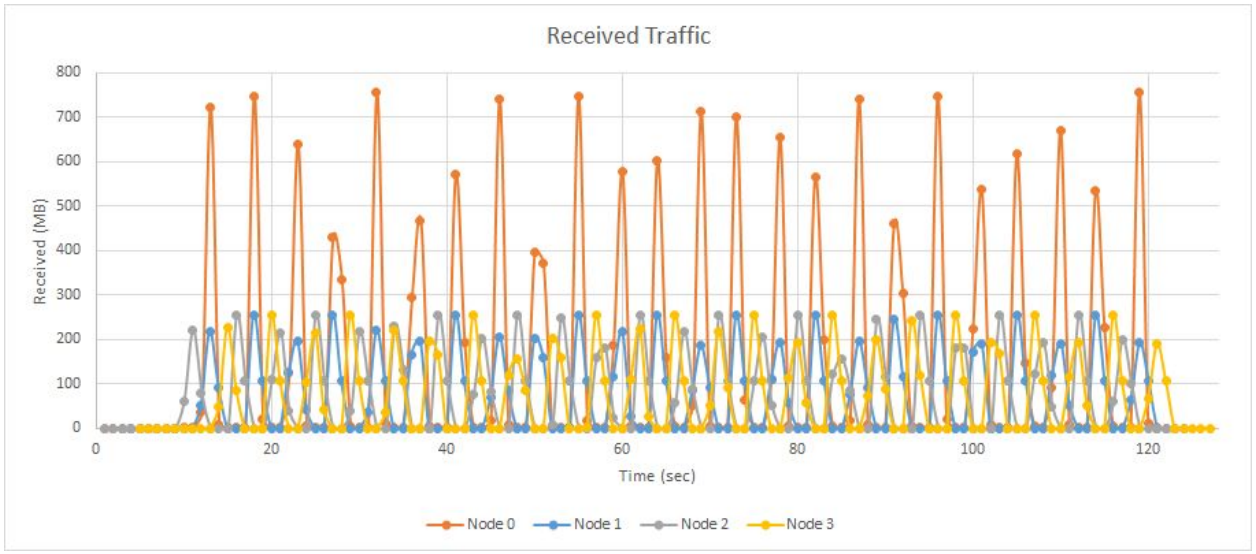
Batch Size	Time Taken (s)
64	75
128	122
256	212

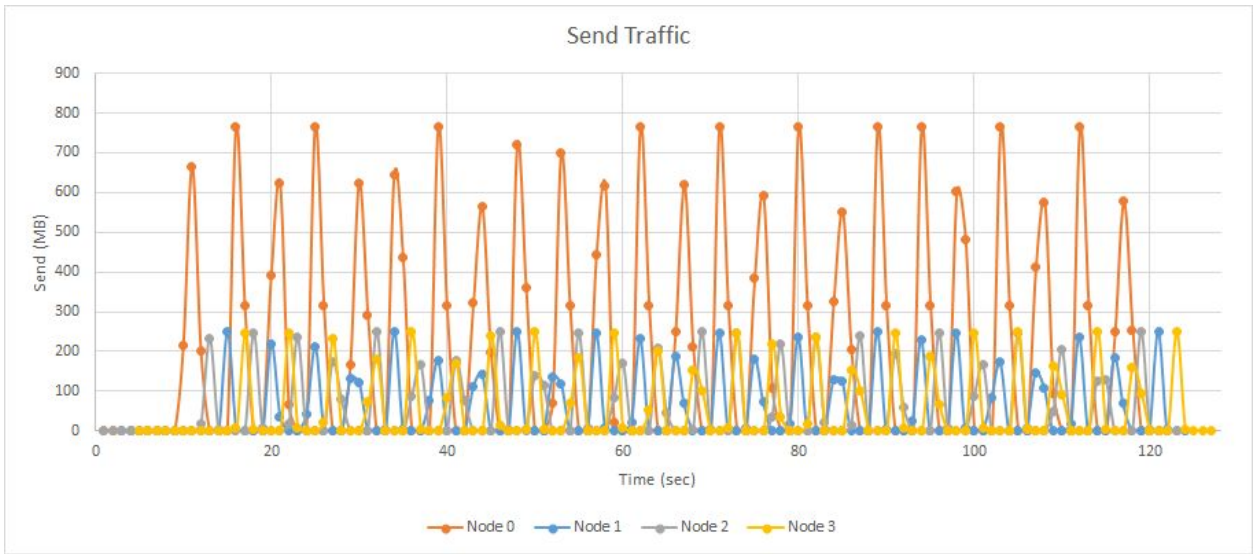
Final loss in all three cases in 7.82 which is achieved just after the 1st iteration.

Time taken in synchronous mode increases with batch size as the parameter server waits for all workers to send updates. Larger batch means larger waiting while the workers complete the task.




CPU Usage
CPU (%)
Time (sec)
Node 0
Node 1
Node 2
Node 3


Received Traffic
Received (MB)
Time (sec)
Node 0
Node 1
Node 2
Node 3


Send Traffic
Send (MB)
Time (sec)
Node 0
Node 1
Node 2
Node 3

Since the process running in synchronous mode, there are alternate cycles of I/O and computation. Hence we see wavelike patterns in all the below graphs.

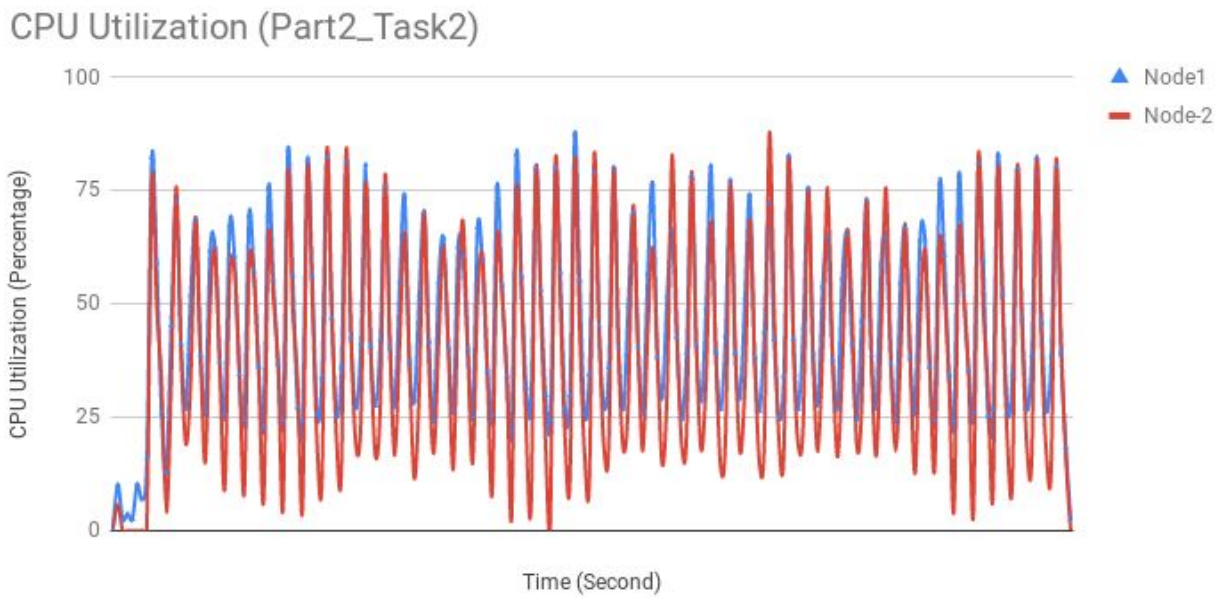
Task 2: 2 Nodes:

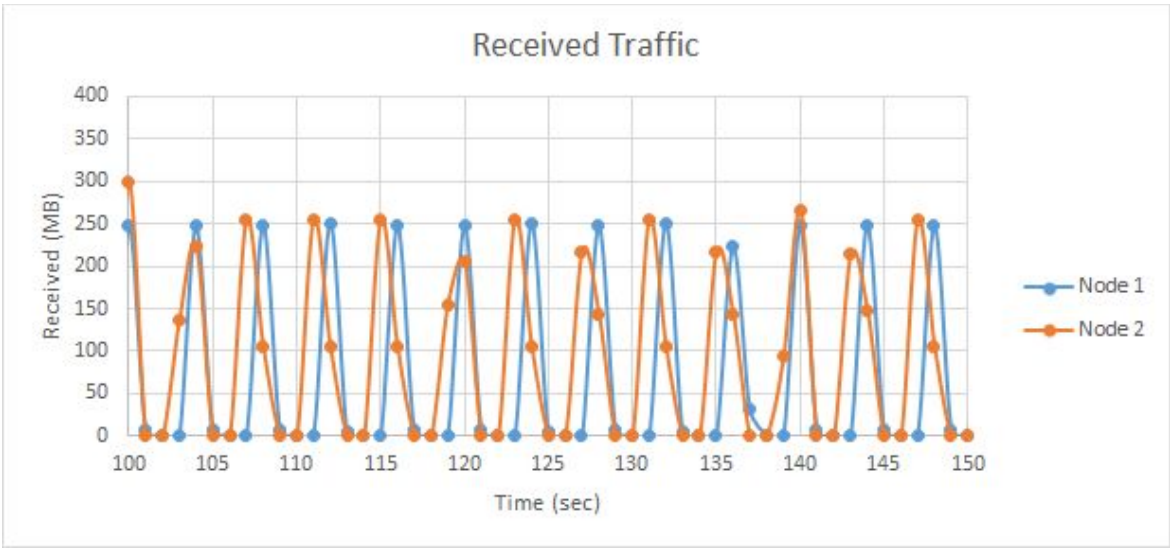
- Time Taken: 217s
- Final loss: 3.91

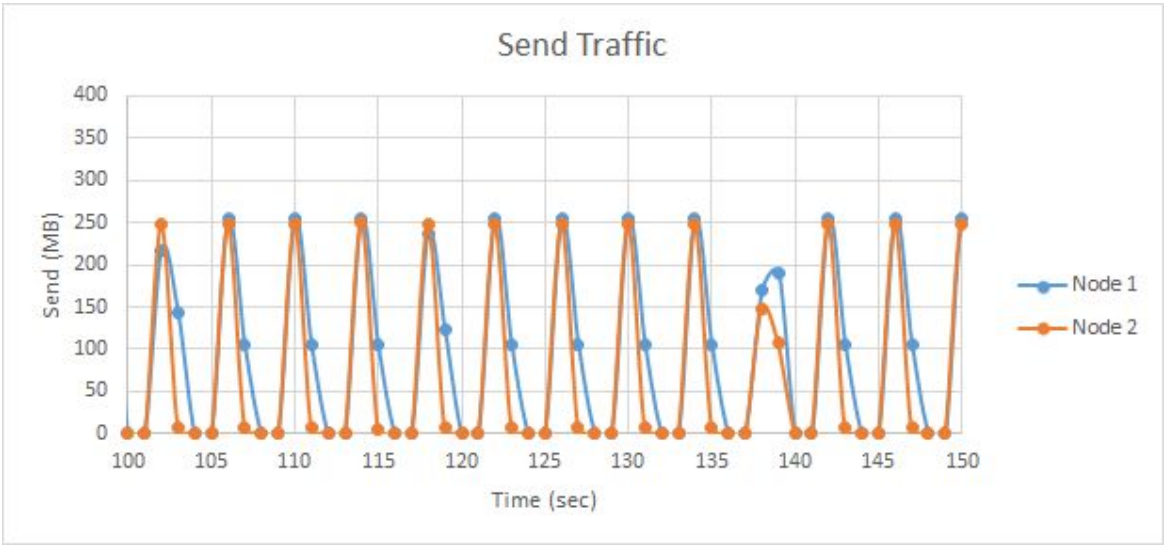
With lesser no. of workers, we lose parallelism to some extent. Hence the time increases.

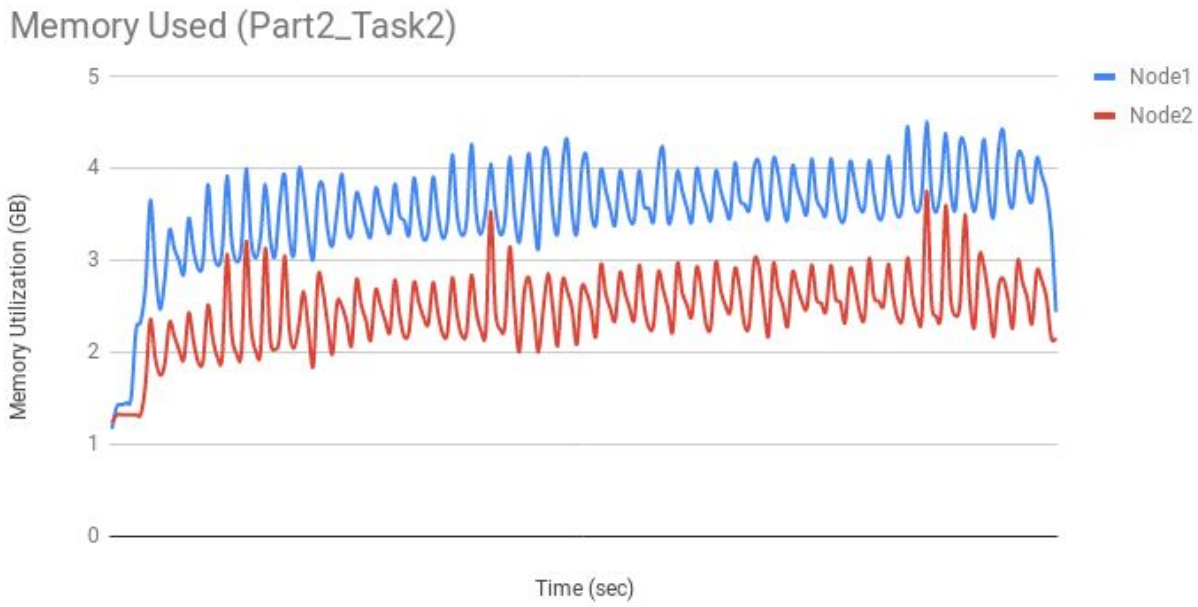
- CPU/Memory/Network Usage

Since the process running in synchronous mode, there are alternate cycles of I/O and computation. Hence we see wavelike patterns in all the below graphs.




Received Traffic
Received (MB)
400
350
300
250
200
150
100
50
0
100
105
110
115
120
125
130
135
140
145
150
Time (sec)
Node 1
Node 2


Send Traffic
Send (MB)
400
350
300
250
200
150
100
50
0
100
105
110
115
120
125
130
135
140
145
150
Time (sec)
Node 1
Node 2



The memory used in node-0 is higher because it also contains the PS server which stores the entire model.

Compared to 2 node case, we see lower network traffic as updates are sent only between two machines. Memory utilization is almost same as model is same.