Report for Assignment 2 : Distributed Deep Learning Training

2017-23643
Taeheon Lee

In this report, I will address analysis on results given from three distributed Deep Learning systems; vanilla-tensorflow, Horovod and Parallax. Before diving into analytic part of this report, I want to **acknowledge** some issues. I and Jaehun Shim, my project teammate, shared a lot of knowledge, experiences for constructing distributed processing environment, coding skills for tensorflow and understandings on Horovod and Parallax. It is due to the situation that we both lack sufficient hardware resources for setting distributed environment and experience for tensorflow.

Deep Learning Model

My work is based on **DeepFam**, a CNN based network for modelling protein sequence families. It was published in January 2018 on *Bioinformatics*. You might check the paper with following url, https://academic.oup.com/bioinformatics/article/34/13/i254/5045722. DeepFam is a classification model, which gets one-hot encoded protein sequence as inputs and give classification predictions as outputs. I used GPCR protein sequence dataset as a training/testing data.
Codes for assignment2 has been written based on the shared codes given by the author of DeepFam, https://github.com/bhi-kimlab/DeepFam. I restructured the code to fit into distributed deep learning systems.
For the execution for the submission, I notices that the performance of learning algorithm converges only after first epoch of training session. Thus, to see the enhancement pattern of the algorithm more obviously, I changed the training code to check the performance for every training batch. Whole training session consists of 200 training step, which means 200 training batches will be used.

Environment Setting

In this assignment, I used one machine with Ubuntu 16.04. My machine is only equipped with CPUs.

Code Structure

- *run_tf.py* : run model with vanilla tensorflow distributed system
- *run_horovod.py* : run model with Horovod
- *run_parallax.py* : run model with Parallax
- *download_dataset.sh* : code for download dataset I have used

Dataset Preparation

As the dataset is not that big, I put dataset file, which consists of ~80 subfamilies of GPCR protein sequences, in submission github. However, if you have difficulty accessing this dataset,

1. create directory name ‘data’
2. in directory ‘data’, execute following command
    A. wget *https://github.com/leetae0130/bd18f-Taeheon_Lee/tree/master/hw2_parallax/data/train.txt*
    B. wget *https://github.com/leetae0130/bd18f-Taeheon_Lee/tree/master/hw2_parallax/data/test.txt*

Commands for Execution

Vanilla Tensorflow Distributed

PS

- python run_tf.py --ps_hosts localhost:12345 --worker_hosts localhost:12346 localhost:12347 --job_name ps --task_index 0

Worker

- python run_tf.py --ps_hosts localhost:12345 --worker_hosts localhost:12346 localhost:12347 --job_name worker --task_index 0
- python run_tf.py --ps_hosts localhost:12345 --worker_hosts localhost:12346 localhost:12347 --job_name worker --task_index 0

Horovod

- mpirun --mca btl_vader_single_copy_mechanism none --allow-run-as-root -bind-to none -map-by slot -mca orte_base_help_aggregate 0 -x NCCL_DEBUG=INFO -np 2 -H localhost:2 python run_horovod.py

Parallax

- python run_parallax.py

Correctness of Execution

To check whether the training algorithm produces same result or not, I made a script to get the checkpoint value and compares two different schemes. The name of the script is **checkpoint_compare.sh**

the usage is as follows

```
>> ./checkpoint_compare.sh ~/jay/parallax parallax_ckpt horovod_distributed 100 100
```

$0 checkpoint_compare.sh : name of script

$1 ~/jay/parallax : path to parallax home **(you should modify it!)**

$2 parallax_ckpt : first scheme to put in

$3 horovod_distributed : second scheme to put in

$4 100 : checkpoint number of first scheme

$5 100 : checkpoint number of second scheme

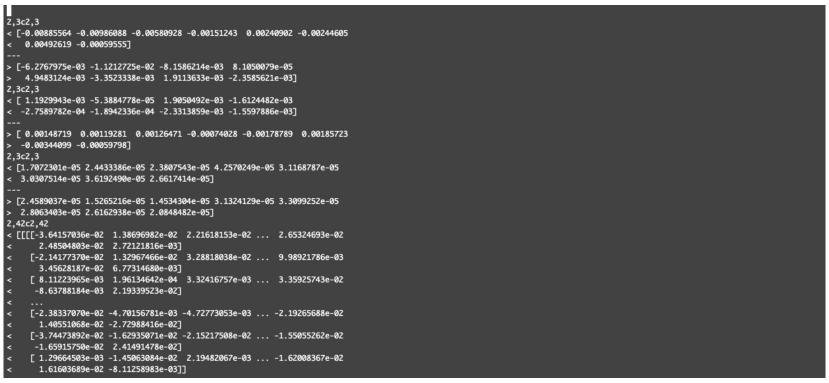
List of schemes : [parallx_ckpt   vanilla_distributed   horovod_distributed]

List of checkpoints : [0   100   200]

(for vanilla distributed, you should put 199 instead of 200)

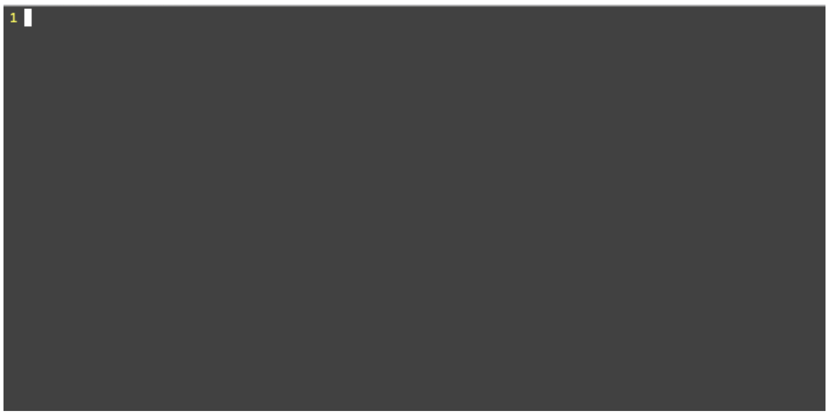
Output comparison file : compare_$2_$2

Result1 : parallax_ckpt vs vanilla_distributed



```
2,3c2,3
< [-0.00885564 -0.00986088 -0.00580928 -0.00151243  0.00240902 -0.00244605
<   0.00492619 -0.00059555]
---
> [-6.2767975e-03 -1.1212725e-02 -8.1586214e-03  8.1050079e-05
>   4.9483124e-03 -3.3523338e-03  1.9113633e-03 -2.3585621e-03]
2,3c2,3
< [ 1.1929943e-03 -5.3884778e-05  1.9050492e-03 -1.6124482e-03
<  -2.7589782e-04 -1.8942336e-04 -2.3313859e-03 -1.5597886e-03]
---
> [ 0.00148719  0.00119281  0.00126471 -0.00074028 -0.00178789  0.00185723
>  -0.00344099 -0.00059798]
2,3c2,3
< [1.7072301e-05 2.4433386e-05 2.3807543e-05 4.2570249e-05 3.1168787e-05
<  3.0307514e-05 3.6192490e-05 2.6617414e-05]
---
> [2.4589037e-05 1.5265216e-05 1.4534304e-05 3.1324129e-05 3.3099252e-05
>  2.8063403e-05 2.6162938e-05 2.0848482e-05]
2,42c2,42
< [[[[-3.64157036e-02  1.38696982e-02  2.21618153e-02 ...  2.65324693e-02
<      2.48504803e-02  2.72121816e-03]
<    [-2.14177370e-02  1.32967466e-02  3.28818038e-02 ...  9.98921786e-03
<      3.45628187e-02  6.77314680e-03]
<    [ 8.11223965e-03  1.96134642e-04  3.32416757e-03 ...  3.35925743e-02
<     -8.63788184e-03  2.19339523e-02]
<    ...
<    [-2.38337070e-02 -4.70156781e-03 -4.72773053e-03 ... -2.19265688e-02
<      1.40551068e-02 -2.72988416e-02]
<    [-3.74473892e-02 -1.62935071e-02 -2.15217508e-02 ... -1.55055262e-02
<     -1.65915750e-02  2.41491478e-02]
<    [ 1.29664503e-03 -1.45063084e-02  2.19482067e-03 ... -1.62008367e-02
<      1.61603689e-02 -8.11258983e-03]]
```

Result2 : parallax_ckpt vs horovod_distributed

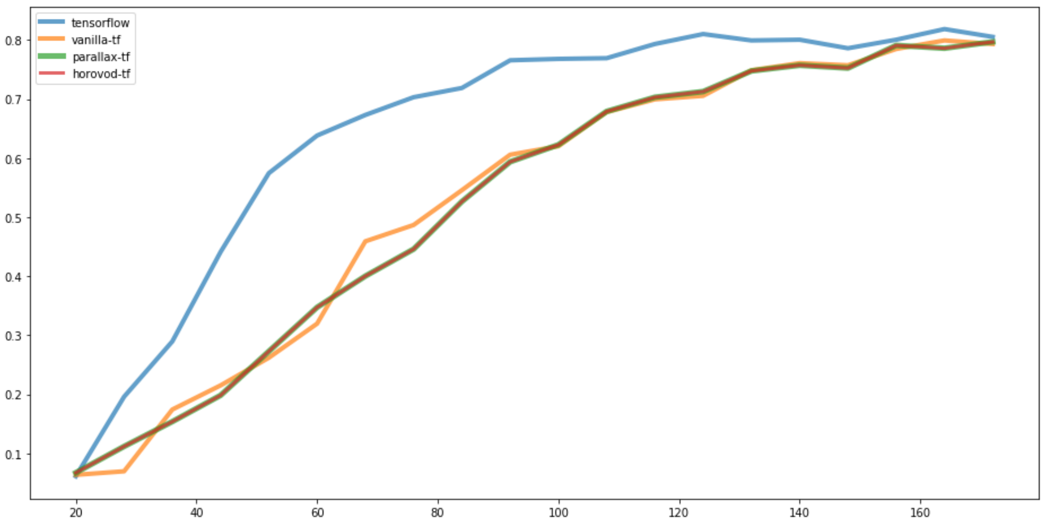


It is noticeable that parallax and horovd gives same output that all the parameter values are equal. However, distributed vanilla tensorflow gives totally different parameter estimations. It is due to the way of incorporating parameter updates from other workers. In vanilla tensorflow I used synchronous replicas updates, where workers waits for all the workers to finish current timestep and updates of parameters are done simultaneously. However, Horovod and Parallax uses rather other schemes where we don't make workers wait for other workers for given timestep. This leads to different update values.

Results Evaluation

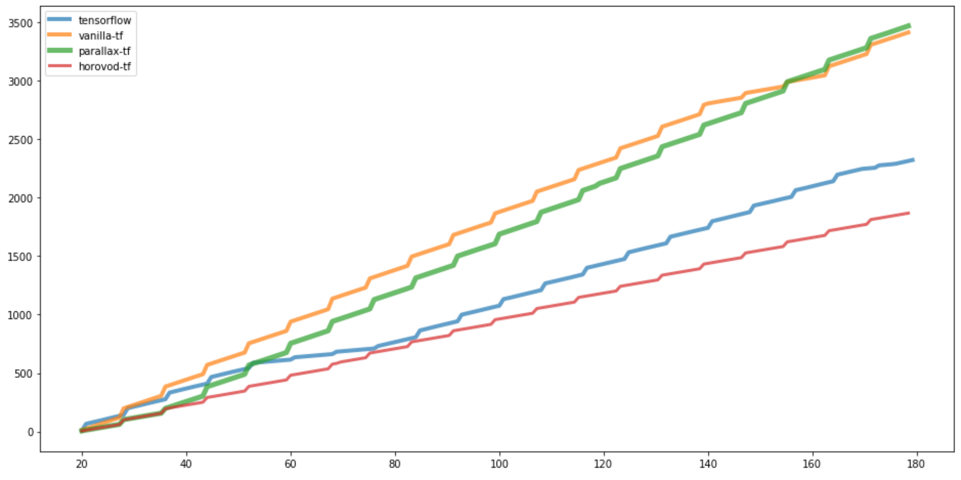
To evaluate performance of three distributed deep learning schemes more fairly, I remove all the random behaviors in the code by setting random seed and discard shuffling behaviors. After executing above command lines, you can will get directory name *logs* and there will be four directories for each distributed training schemes. There will be *log.txt* file which logs the behavioral improvement of the algorithm.

Following graph shows accuracy improvement of each scheme.



It seems that there is a significant difference between non-distruted session and distributed session, non-distributed session shows faster improvement in early stages. The reason for such phenomena is that non-distributed session can fully utilize the data as they update parameters one after one. However, in distributed session, some informations in data updates might be discarded in the process of integration.

Next plot shows the time spent until the timestep of training session. As my testing environment only contains one machine, I think fair and robust comparison has not been done. However, it seems obvious that Horovoid is much more optimized than other schemes. I acknowledge the insufficient resources might lead to wrong results.


tensorflow
vanilla-tf
parallax-tf
horovod-tf