ETH zürich

CSCS

Scalability of Deep Learning Frameworks

Sumeet Gyanchandani

Contents

1. Introduction

Deep Neural Networks are making breakthroughs in a wide variety of everyday technologies, such as speech recognition, image recognition, computer vision, machine translation, and many others. With these recent achievements of Deep Learning, the field has received a lot of acceptance into various fields and is being used in tasks ranging from cooling the data centers at Google, driving cars, healthcare and even to take investment decisions. As the problem size is getting larger by the day, we are witnessing a surge in demand for computational power. Fortunately, we are not restricted to a single machine and all the major deep learning frameworks are making constant efforts towards making distributed training of neural networks faster and more efficient.

We started this project by performing a broad study on the HPC readiness of many open source deep learning frameworks. We then cherry-picked the most promising ones that use MPI backends and GPUs in order to evaluate them on Piz Daint. Next, we speak about the current state of distributed training in Facebook's Caffe2 and it's comparison with the gold standard of deep learning frameworks - Google's TensorFlow. Lastly, we present a comparison of the performance of Horovod, an MPI-based open source communication framework by Uber, against the in-house Cray PE ML Plugin in training Inception v3 using TensorFlow.

2. Environment

Piz Daint is a hybrid Cray XC40/XC50 supercomputer at CSCS. The system has Aries routing and communications ASIC, with Dragonfly network topology [1].

At the time of writing, it is the third most powerful supercomputer in the world [2] and in the top ten of the most energy-efficient supercomputers [3]. Each GPU node that we use on Piz Daint is equipped with an NVIDIA Tesla P100 [4].

3. HPC readiness of Deep Learning Frameworks

We started this project by performing a study on the HPC readiness of the Deep Learning Frameworks. The entire study is depicted in Table 1 on the next page. The frameworks that have been highlighted in green are the ones that we selected for evaluation during this project, viz, **TensorFlow 1.10.0** and **Caffe2**. And those in purple are the ones that we recommend for future evaluation, viz., **Chainer, Pytorch 1.0, TensorFlow 2.0, PaddlePadldle, and Microsoft Cognitive Toolkit**.

4. TensorFlow vs Caffe2

4.1 Motivation

We had no doubt that we need to evaluate TensorFlow's performance on Piz Daint. Over the past 2 years, TensorFlow has evolved to become the gold standard of the Deep Learning Frameworks. We started to look for a worthy contender to put against TensorFlow and understand how they are different from each other.

Few months ago, Facebook announced that they have trained ResNet-50 on ImageNet data with a minibatch size of 8192 on 256 GPUs in just 1 hour with their Deep Learning Library - Caffe2 [5]. We immediately got what we needed and we started to build Caffe2 on Piz Daint.

Author	Framework	Distributed Communication Mechanism	Link
Google	Tensorflow 1.10	Parameter Server + MPI allreduce using Uber's Horovod https://eng.uber.com/horovod/	https://www.tensorflow.org/deploy/distributed
Facebook	Caffe2	MPI rendezvous using Facebook's Gloo https://caffe2.ai/docs/distributed-training.html	https://caffe2.ai
Google	Tensorflow 2.0	Launches in 2019. High expectation of MPI rendezvous still working	https://www.tensorflow.org/deploy/distributed
Facebook AI Research, Twitter, Uber, NVIDIA and many others	Pytorch 1.0 (Pytorch + Caffe2)	Launches in 2019. MPI rendezvous using Gloo https://github.com/facebookincubator/gloo	https://pytorch.org/2018/05/02/road-to-1.0.html
Apache	MXNet	No MPI rendezvous yet. But found a feature request! The progress can be tracked here: https://github.com/apache/incubator-mxnet/pull/10696	https://mxnet.apache.org
Open Source	Keras	High-level deep learning API which works on top of TensorFlow, Microsoft Cognitive Toolkit and Theano. Uses Distributed backend from them.	https://keras.io
MILA	Theano	An implementation exists on mpi4py, can perform multi-node multi-gpu distribution using MPI https://github.com/uoguelph-mlrg/Theano-MPI	http://deeplearning.net/software/theano/
Microsoft	Cognitive Toolkit (CNTK)	HPC ready! It also has examples for multi-node and multi-GPUs training. https://docs.microsoft.com/en-us/cognitive-toolkit/multiple-gpus-and-machines	https://www.microsoft.com/en-us/cognitive-toolkit/
IBM	Watson Machine Learning	It is a cloud-based enterprise platform similar to AWS	https://www.ibm.com/cloud/machine-learning
CMU (Acquired by Apple)	Turi	No MPI rendezvous yet	https://turi.com
Preferred Networks	Chainer	Have developed a package for Distributed Deep Learning called ChainerMN. Trained ImageNet dataset on ResNet-50 network in 15 minutes, which is four times faster than the previous record held by Facebook. https://github.com/chainer/chainermn	https://chainer.org
Amazon	DSSTNE	Does not have multi-node support	https://github.com/amzn/amazon-dsstne
H2O.ai	H2O	Has a distributed product called Deep Water, which doesn't seem to use MPI, probably uses shared location for communication. Poorly documented! https://github.com/h2oai/deepwater	https://www.h2o.ai
Facebook, Google Deepmind	Torch	An implementation by Facebook AI Research that can perform multi-node multi-gpu distribution using MPI. https://github.com/facebookresearch/TorchMPI	http://torch.ch
CMU	DyNet	This is a pull request that successfully provided Multi-node support to DyNet. Doesn't seem to use MPI though https://github.com/clab/dynet/pull/704	http://dynet.io
Artelnics	OpenNN	Uses MPI	http://www.opennn.net
Apache	SINGA	Uses MPI! https://svn.apache.org/repos/infra/websites/production/singa/content/v0.1.0/communication.html	https://singa.apache.org/en/index.html
Hao Dong	TensorLayer	Another Wrapper Library on top of TensorFlow. Supports MPI. https://github.com/tensorlayer/tensorlayer/tree/master/examples/distributed_training	https://tensorlayer.readthedocs.io/en/stable/
Baidu	PaddlePaddle	Uses Parameter Server and MPI rendezvous https://github.com/PaddlePaddle/Paddle	http://www.paddlepaddle.org/en

Table 1. Study of HPC readiness of Deep Learning Frameworks

4.2 Building Caffe2 on Piz Daint

The first task that we did was to build Caffe2 and it's dependencies for a single node of Piz Daint. We got success in this task with very little changes and lesser time than we had expected. Next, we wanted to build Caffe2 with MPI so that it can distribute its training over multiple nodes of Piz Daint. We got an EasyBuild configuration file [6] for this task from the internet but it failed miserably.

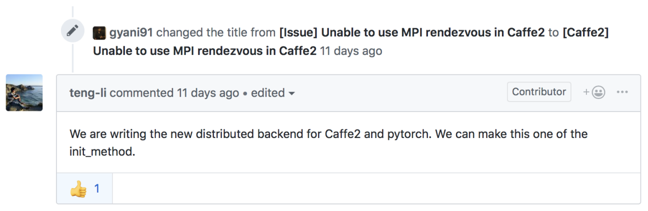
The initial problem that we noticed was that most of the Deep Learning Frameworks are compiled using OpenMPI and Piz Daint uses MPICH. We changed this configuration wherever required and compiled Caffe2 natively on Piz Daint but it still didn't run successfully.

Later we discovered that the communication framework - **Gloo**, that Caffe2 was using for its distributed training was also built using OpenMPI. Hence, we built Gloo natively with MPICH and built Caffe2 with MPICH support on top of it. Even this endeavor failed. After changing several configurations and part of the Gloo's code, we had hit a dead end on a runtime error, to be specific a segmentation fault error (SIGSEGV) which was sourced from Gloo.

Speculating that this error may be caused by version incompatibilities on Piz Daint, we decided to create a docker image for Caffe2 with MPICH support and use shifter to run it on Piz Daint. We got the same SIGSEGV error that we were stuck on when we built it natively on Piz Daint. The Easybuild config files and Dockerfiles are present in the Resources folder of the git repository and the procedures to run them are given in Appendix A & B.

4.3 Conclusions regarding distributed Caffe2

As we were getting the same runtime error no matter what configuration we tried, we decide to raise a case with the Caffe2 developers on their Github account [7]. One of the prominent contributors from Facebook AI Research replied with:


gyani91 changed the title from **[Issue] Unable to use MPI rendezvous in Caffe2** to **[Caffe2] Unable to use MPI rendezvous in Caffe2** 11 days ago

teng-li commented 11 days ago • edited

Contributor

We are writing the new distributed backend for Caffe2 and pytorch. We can make this one of the init_method.

👍 1

which goes to say that this can not be achieved by using the current Caffe2 framework and that you have to wait for an unknown amount of time in order to be able to run distributed Caffe2 on Piz Daint.

This made us draw two important conclusions regarding the state of distributed training in Caffe2:

1. Caffe2 has poor version control. TensorFlow has gone through 11 minor version changes from TensorFlow 1.0 to 1.10 in a matter of $1\frac{1}{2}$ years. While Caffe2 neither has any sub versioning nor it has any **pip packages** that you can install. You always have to **git clone --recursive** and then make it using cmake. It works somedays, and on other days you have to find git checkouts from a few days back in order to have a buildable version.

2. Facebook has just announced that they would be combining both their deep learning libraries *Caffe2* and *Pytorch* to a single library called **Pytorch 1.0** [8]. The fast changing source code and the improper version control are the major reasons for the distributed backends of both these libraries not currently working.

4.4 Running benchmarks on a single node

The problems mentioned in the previous section left us with no choice but to run the benchmarks on the single node of Piz Daint. We took the classical Image Classification example in order to test these Deep Learning Frameworks. The metric on which we would be judging the performance of these frameworks is images/sec. Moreover, learning that Pytorch is the next big thing, we also build a container for it and included it as a part of our benchmark.

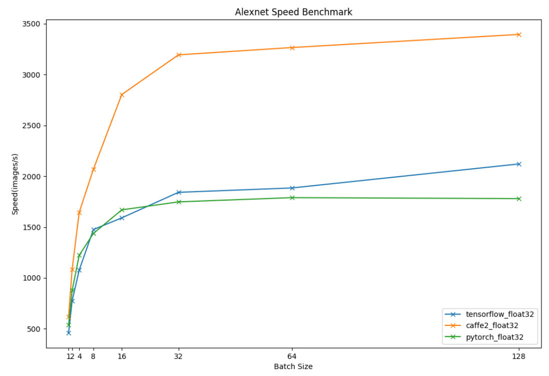


Figure 1. Speed benchmark for Alexnet

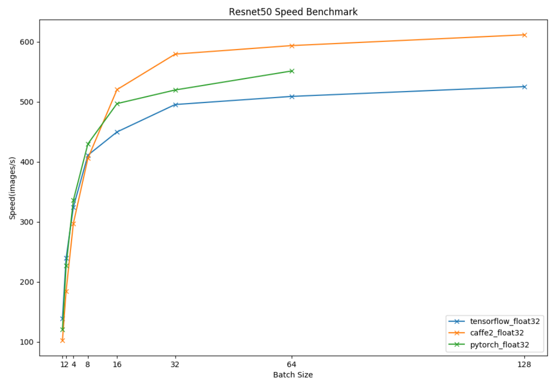


Figure 2. Speed benchmark for Resnet50

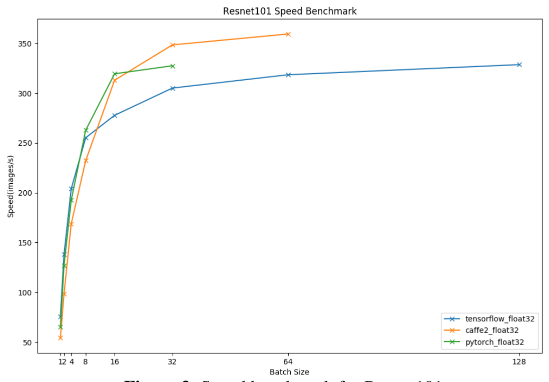


Figure 3. Speed benchmark for Resnet101

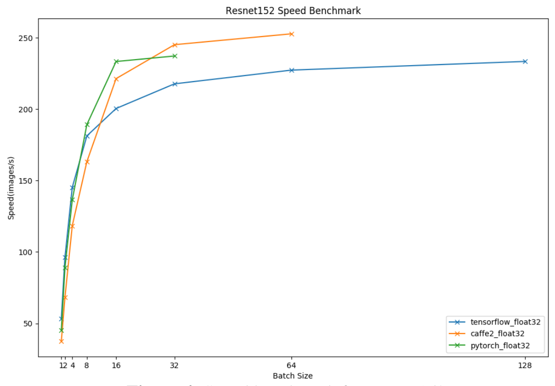


Figure 4. Speed benchmark for Resnet152

Figure 1-4 depict the benchmark of TensorFlow, Caffe2, and Pytorch in training various convolutional neural networks for different minibatch sizes. The neural nets being used are Alexnet, Resnet50, Resnet101, and Resnet152.

We observed that Pytorch was not able to complete the training of Resnet50 with a minibatch size of 128. For larger neural nets like Resnet101 and Resnet152, both Caffe2 and Pytorch were not able to complete the training for 64 and 32 minibatch sizes respectively.

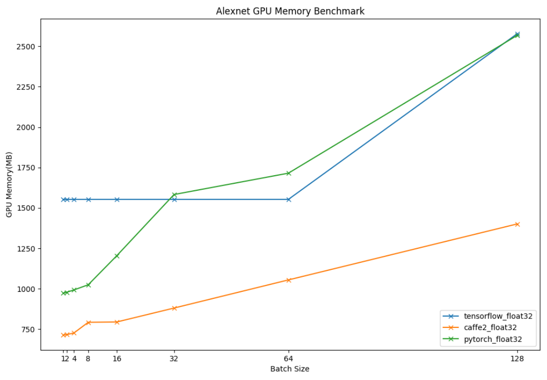


Figure 5. Memory benchmark for Alexnet

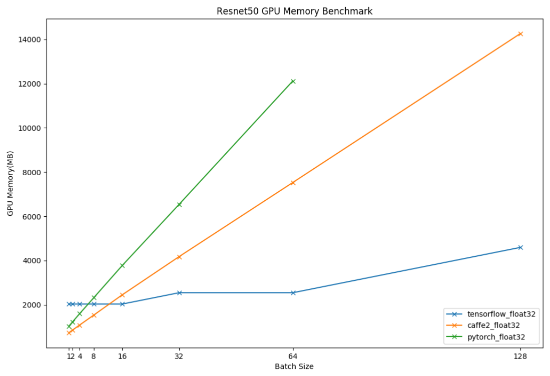


Figure 6. Memory benchmark for Resnet50

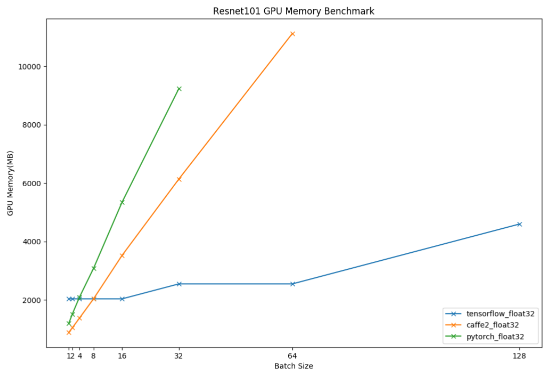


Figure 7. Memory benchmark for Resnet101

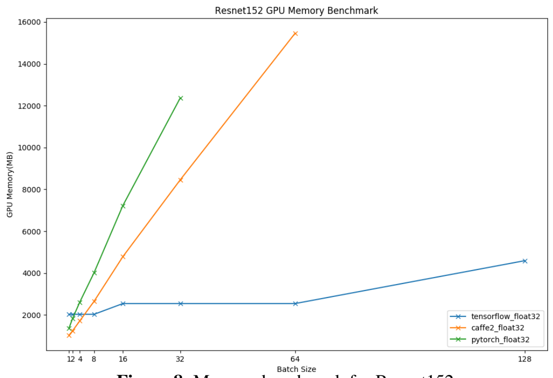


Figure 8. Memory benchmark for Resnet152

To investigate why this might have happened we performed a memory benchmark for the training. Figure 5-8 depict the memory required by TensorFlow, Caffe2, and Pytorch in training these neural networks for different minibatch sizes. We observed that Caffe2 and Pytorch are very resource hungry and that TensorFlow does a great job at memory management. Caffe2 and Pytorch were not able to complete the training at 32 and 64 minibatch sizes because they exhausted the available 16 GB memory of the GPU. Due to this overhead of memory management, TensorFlow is marginally slower than Caffe2 and Pytorch. The procedure to run these benchmarks are given in Appendix C.

5. Cray PE ML Plugin vs Horovod

In this section, we discuss another latest framework. This time it is a communication framework by Uber called Horovod [9]. At CSCS, the current communication framework that is being used is Cray PE ML Plugin [10]. It is a proprietary software that supports only TensorFlow 1.3.0. Using it requires some amount of training, as you have to modify your code at several places.

Horovod is an open source framework that supports TensorFlow 1.10.0, latest versions of Keras and Pytorch. Writing distributed code is easy due to decent support from the deep learning frameworks and good example base. For instance, in order to run the benchmarks for this report, all we had to do is download the standard tf_cnn_benchmark scripts from TensorFlow [11] and change the **--variable_update** parameter to **horovod**. All the other changes to the benchmark scripts and the library files were done by the TensorFlow team.

The primary motivation behind choosing to evaluate the performance of Horovod was to find an open source alternative to the otherwise paid service from Cray. Figure 9 shows the performance of Horovod against Cray PE ML Plugin in training Google's Inception v3 using TensorFlow on Piz Daint.

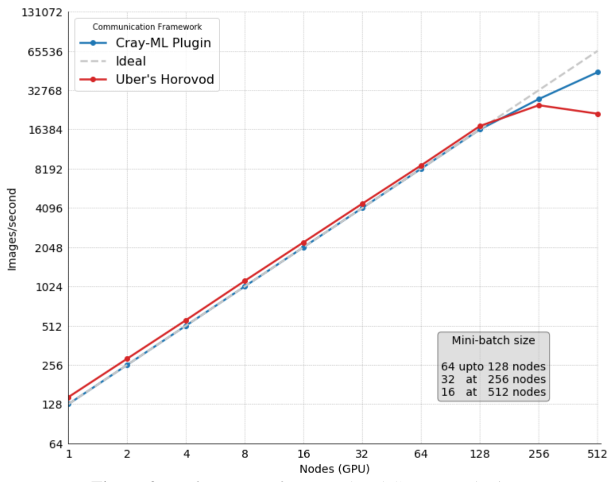


Figure 9. Performance of Horovod and Cray ML Plugin I

There are two things worth noticing here:

1. The performance of Horovod on a single node is higher than that of Cray PE ML Plugin. We attribute that to the higher TensorFlow version, and not Horovod itself.

2. For some unjustified reasons, Cray professionals recommended reducing the minibatch size to half at 256 nodes and to a quarter at 512 nodes. As we expected, Horovod scaled properly up to 128 nodes and reduced at 256 and 512 nodes but somehow Cray's framework performed better than Horovod at these configurations.

We did not understand the motivation behind reducing the minibatch size abruptly at these points and how Cray was delivering the same scalability on half the minibatch size. Hence, we decide to run the scalability benchmark again, comparing Cray PE ML Plugin with their recommended configurations vs a constant minibatch size of 64 for Horovod. The result was exactly what we expected - perfect scaling!

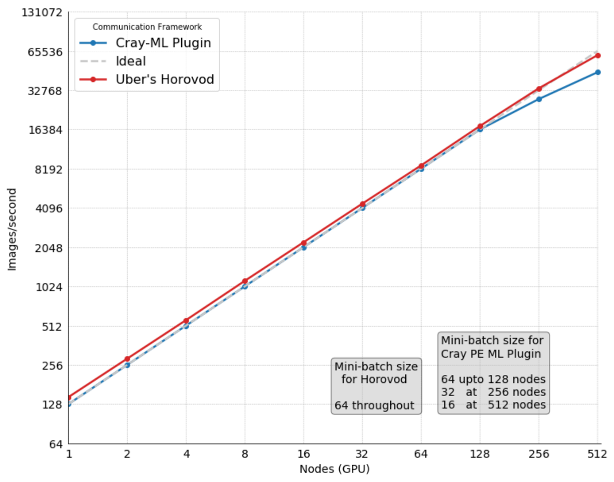


Figure 10. Performance of Horovod and Cray ML Plugin II

The procedure to build Horovod in given in Appendix D and the procedure to run the benchmarks in given in Appendix E.

6. Impact on accuracy while scaling

What we have discussed so far in this report is the HPC perspective of scalability of deep learning frameworks. There is another perspective that we care about, that is the Machine Learning perspective. We would also want to evaluate the impact of scaling on the accuracy and the convergence of the model. We were able to lay the foundation for this on Piz Daint and we would speak about it in this section.

6.1 TensorBoard

The deep neural networks that we train can get quite complex and confusing. Google has designed a suite of visualization tools called TensorBoard [12]. It is a great tool for visualizing the computational graph and training statistics during and after the training.

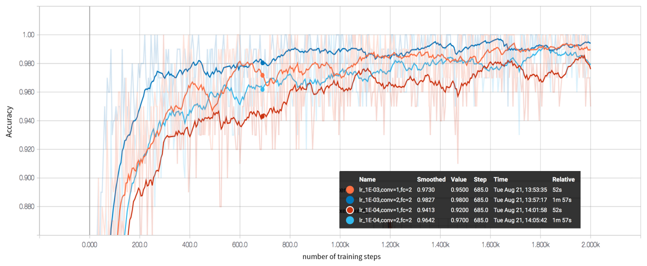
What we discovered is that even other deep learning frameworks like Caffe2, Pytorch, and Microsoft Cognitive Toolkit have provided a way to dump their accuracy test results during the training (and many other summaries) into the disk, which can be later visualized using TensorBoard.

6.2 TensorBoard support for Caffe2

In order to be able to compare the impact on the accuracy during distributed training in TensorFlow and Caffe2, we made a Docker container for Caffe2 with c2board [13], which helps Caffe2 to dump its computational graph and training statistics in TensorBoard compatible format (protobuf). The procedure for building Caffe2 with c2board support in given in Appendix F.

6.3 Accuracy on a single node of Piz Daint

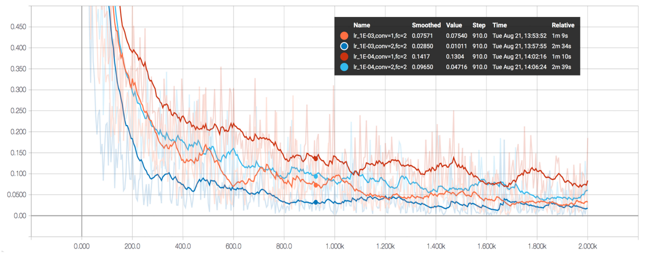
Figure 11 shows the progression of accuracies of four different neural networks at different training steps on a single node of Piz Daint using TensorFlow.



	Name	Smoothed	Value	Step	Time	Relative
●	lr_1E-03,conv=1,fc=2	0.9730	0.9500	685.0	Tue Aug 21, 13:53:35	52s
●	lr_1E-03,conv=2,fc=2	0.9827	0.9800	685.0	Tue Aug 21, 13:57:17	1m 57s
●	lr_1E-04,conv=1,fc=2	0.9413	0.9200	685.0	Tue Aug 21, 14:01:58	52s
●	lr_1E-04,conv=2,fc=2	0.9642	0.9700	685.0	Tue Aug 21, 14:05:42	1m 57s

Figure 11. Accuracy on a single node of Piz Daint

Figure 12 shows the progression of cross-entropy loss for these neural networks at different training steps. In future, we wish to evaluate the impact on these accuracies when we scale from a single node to 1024 or even 2048 nodes.



	Name	Smoothed	Value	Step	Time	Relative
●	lr_1E-03,conv=1,fc=2	0.07571	0.07540	910.0	Tue Aug 21, 13:53:52	1m 9s
●	lr_1E-03,conv=2,fc=2	0.02850	0.01011	910.0	Tue Aug 21, 13:57:55	2m 34s
●	lr_1E-04,conv=1,fc=2	0.1417	0.1304	910.0	Tue Aug 21, 14:02:16	1m 10s
●	lr_1E-04,conv=2,fc=2	0.09650	0.04716	910.0	Tue Aug 21, 14:06:24	2m 39s

Figure 12. Loss on a single node of Piz Daint

7. Conclusion

1. Caffe2 is faster than TensorFlow (at least on a single node) but at the same time it is very resource greedy and has a much higher memory requirement than that of TensorFlow

2. Distributed training in Caffe2 and Pytorch are not worth exploring until we get proper benchmarks from Facebook AI Research after they have launched Pytorch 1.0

3. Getting these benchmarks still doesn't guarantee that Pytorch 1.0 would work with MPICH.

4. Horovod is a great alternative to Cray PE ML Plugin. It has the support of major Deep Learning Frameworks. It is open source, which means it would be rigorously used and well contributed to.

8. Project Repository

The resources and the results for this project are available on Scalability of Deep Learning Frameworks

References

[1] Piz Daint at Swiss National Supercomputing Centre, https://www.cscs.ch/computers/piz-daint/

[2] The top 500 list, https://www.top500.org/lists/2017/11/

[3] The green 500 list, https://www.top500.org/green500/lists/2017/11/

[4] NVIDIA Tesla P100, https://www.nvidia.com/en-us/data-center/tesla-p100/

[5] P. Goyal, P. Dollàr, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, K. He, Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, Apr 2018, arXiv:1706.02677

[6] M. Schoengens, G. P. Pezzi, Caffe2 easybuild config file, https://github.com/eth-cscs/production/blob/master/easybuild/easyconfigs/c/Caffe2/Caffe2-20180106-CrayGNU-17.08-cuda-8.0-python3.eb

[7] Unable to use MPI rendezvous in Caffe2, https://github.com/pytorch/pytorch/issues/10582

[8] Facebook announces PyTorch 1.0, a more unified AI framework, https://techcrunch.com/2018/05/02/facebook-announces-pytorch-1-0-a-more-unified-ai-framework/

[9] Meet Horovod: Uber's Open Source Distributed Deep Learning Framework, https://eng.uber.com/horovod/

[10] Run TensorFlow with the Cray PE Machine Learning Plugin, https://pubs.cray.com/content/S-2589/1.1.UP00/xctm-series-urika-xc-analytic-applications-guide/run-tensorflow-with-the-cray-pe-machine-learning-plugin

[11] High Performance Benchmark Scripts, https://github.com/tensorflow/benchmarks/tree/master/scripts/tf_cnn_benchmarks

[12] TensorBoard: Visualizing Learning, https://www.tensorflow.org/guide/summaries_and_tensorboard

[13] c2board, https://github.com/endernewton/c2board

Appendix

Appendix A: Building Caffe2 using EasyBuild Framework

building:

```bash
git clone https://github.com/eth-cscs/production.git
```

replace */production/easybuild/easyconfigs/c/Caffe2/Caffe2-20180106-CrayGNU-17.08-cuda-8.0-python3.eb* with *Scalability of Deep Learning Frameworks/Resources/Easy build/Caffe2-20180106-CrayGNU-17.08-cuda-8.0-python3.eb*

```bash
export EASYBUILD_PREFIX=$SCRATCH/easybuild/daint
export EB_CUSTOM_REPOSITORY=$SCRATCH/production/easybuild
module load daint-gpu EasyBuild-custom
eb Caffe2-20180106-CrayGNU-17.08-cuda-8.0-python3.eb -r
```

using:

copy *Scalability of Deep Learning Frameworks/Resources/Easy build/module files/caffe2* to your modules folder.
If you do not have the modules folder you can create one by:

```bash
cd $HOME
mkdir modules
cd modules
```

Append **module use $HOME/modules** to your **.bashrc** and restart your terminal or ssh connection. Run *Scalability of Deep Learning Frameworks/Resources/Easybuild/module files/load.sh* and now you can use Caffe2.

Appendix B: Building Caffe2 in a Docker Container

Image creation on local machine:

```bash
cd Scalability\ of\ Deep\ Learning Frameworks/Resources/
     Dockerfiles/caffe2_container
docker build -t caffe2_container .
```

Migrating the image to Piz Daint:

on the local machine do:

```bash
docker save --output caffe2_container.tar caffe_container
```

copy the caffe2_container.tar to Piz Daint and on Piz Daint do:

```bash
module load daint-gpu
module load shifter-ng
srun -C gpu shifter load ./caffe2_container.tar
     caffe2_container
srun -N 1 -n 1 -C gpu shifter run load/library/
     caffe2_container python --version
```

Appendix C: Running Benchmarks on a single node

```bash
cd Scalability\ of\ Deep\ Learning Frameworks/Resources/
     Single\ node\ benchmark
sbatch run_benchmark.sh
```

Appendix D: Building Horovod in a Docker Container

Image creation on local machine:

```bash
cd Scalability\ of\ Deep\ Learning Frameworks/Resources/
     Dockerfiles/horovod_container
docker build -t horovod_container .
```

Migrating the image to Piz Daint:

on the local machine do:

```bash
docker save --output horovod_container.tar
     horovod_container
```

copy the horovod_container.tar to Piz Daint and on Piz Daint do:

```bash
module load daint-gpu
module load shifter-ng
srun -C gpu shifter load ./horovod_container.tar
     horovod_container
srun -N 1 -n 1 -C gpu shifter run load/library/
     horovod_container python --version
```

Appendix E: Running Benchmarks for Horovod

```bash
cd Scalability\ of\ Deep\ Learning Frameworks/Resources/
     Horovod\ benchmark
./run_benchmark.sh
```

you can use *Scalability of Deep Learning Frameworks/Resources/Horovod benchmark/Plots.ipynb* to plot the graphs in Section 5.

Appendix F: Building Caffe2 with TensorBoard support

Image creation on local machine:

```bash
cd Scalability\ of\ Deep\ Learning Frameworks/Resources/
     Dockerfiles/caffe2_tensorboard
docker build -t caffe2_tensorboard .
```

Migrating the image to Piz Daint:

on the local machine do:

```bash
docker save --output caffe2_tensorboard.tar
     caffe2_tensorboard
```

copy the caffe2_tensorboard.tar to Piz Daint and on Piz Daint do:

```bash
module load daint-gpu
module load shifter-ng
srun -C gpu shifter load ./caffe2_tensorboard.tar
     caffe2_tensorboard
srun -N 1 -n 1 -C gpu shifter run load/library/
     caffe2_tensorboard python --version
```