Early Results of Deep Learning on the Stampede2 Supercomputer

Zhao Zhang, Weijia Xu, Niall Gaffney, Daniel Stanzione
Texas Advanced Computing Center
zzhang,xwj,ngaffney,dan@tacc.utexas.edu

ABSTRACT

We present early results of the deep learning work on the Stampede2 supercomputer. Our goal is to enable scalable and efficient deep learning model training and serving to expedite scientific discovery. We build three popular deep learning frameworks, namely, Intel-Caffe, MXNet, and TensorFlow. With the built-in applications of these frameworks (CaffeNet, AlexNet, GoogLeNet, and Cifar10), we measure the scalability in both strong scaling and weak scaling way. At the time of writing, we are able to build and run Intel-Caffe, MXNet, and TensorFlow on multiple KNL nodes. While the MXNet and TensorFlow performance are still being tuned, we manage to scale the afore-mentioned applications in Caffe on 512 KNLs with ~80% efficiency compared to a single KNL performance.

1 INTRODUCTION

Scientists from many domains are actively exploring and adopting deep learning as a cutting-edge methodology to make research breakthrough. The impact of deep learning is across fields, such as drug discovery [4], disease diagnosis [10], particle physics [5], and neurology [20]. We have also observed a new trend of using deep learning model to reduce the simulation complexity, e.g., in condensed matter physics [7]. Domain scientists usually use open source deep learning frameworks to train and serve the deep learning models. Caffe [15], Theano [6], Torch [11], TensorFlow [3], CNTK [23], and MXNet [8] are a few representative frameworks among many. Given the computation, communication, and I/O pattern of the deep learning applications, they are naturally good candidates for supercomputers [9].

We, the Texas Advanced Computing Center (TACC), are embracing this new type of application and the programming paradigm change it may bring. Our present roadmap to support deep learning at TACC is to first enable scalable and efficient model training with a few popular open source deep learning frameworks, then to open the access to our users on the Stampede2 supercomputer [21]. At the time of writing, we manage to build and run Intel-Caffe [13], MXNet, and TensorFlow on multiple KNL nodes. We complete a comprehensive performance measurement of Intel-Caffe using existing image classification applications of CaffeNet [1], AlexNet [18], and GoogLeNet [22] on ImageNet dataset [12] and ConvNet [2] on the Cifar10 dataset [17].

We tune the Intel-Caffe performance on a single KNL, and determine 64 OpenMP threads reaches the peak performance. The 64-thread single node performance is ~2x faster than a Nvidia's K40 GPU and ~2x slower than that of a P100 GPU. Our strong scaling performance shows a ~50% efficiency on 8 KNLs for various applications. And the weak scaling performance shows a ~80% efficiency up to 512 KNLs.

2 ACCOMPLISHMENTS

Early results of TACC's deep learning work include building Intel-Caffe, MXNet, and TensorFlow on the Stampede2 supercomputer, single node performance evaluation and comparison to Nvidia GPUs, and the scale-out performance measurements.

The Stampede2 supercomputer is equipped with 4,200 KNL compute nodes in Phase 1, with 96 GB DDR4 and 16 GB MCDRAM. The interconnect is a 100 Gb/sec Intel Omni-Path network with a fat-tree topology. The file system used in this report is a Lustre deployment with ~30 PB storage capacity.

The Intel distribution of Caffe (the Jan 2017 release) is compiled with Intel Machine Learning Scaling Library (MLSL) [14]. For MXNet and TensorFlow, we use the v0.10.0-79-g790328f tag and v1.3.0-rc2 release, respectively. All three frameworks support parallel training enabled by MPI. In particular, Intel-Caffe uses the data-parallel approach for distributed training. In addition to the data-parallel approach, MXNet, and TensorFlow also support the model-parallel approach in the recursive neural network, where the memory consumption of parameters exceeds a single node's memory space.

We use four applications: ConvNet on Cifar10 dataset, and CaffeNet, AlexNet, and GoogLeNet on a 100-category ImageNet dataset. All these four applications are Convolutional Neural Networks.

For each data point, we run the experiment three times and report the average and standard deviation. The only exceptions are in the 512-node and 1024-node measurements, where we only measure once due to the expensive node hour cost.

2.1 Intel-Caffe Single Node Performance

To tune the Intel-Caffe performance on a single node, we use two applications: ConvNet on the Cifar10 dataset and CaffeNet on a 100-category ImageNet dataset. There are a few runtime options in the Intel-Caffe command line. Users can specify the engine to be MKL2017, MKLDNN, or leave it blank to use the default option. Our observation is that MKL2017 run ~3.7x and ~1.4x faster than the default option and using the MKLDNN option outputs inconsistent results.

Next, we fix the problem size and measure the performance with varying the number of OpenMP threads. As shown in Figure 1, using 64 OpenMP threads results in best performance. Fewer threads, e.g., 32, can not make efficient use of the many-core architecture, and 128 threads saturate the processor and slow down the application performance. So, in the rest of the experiments, we fix the number of OpenMP threads to be 64.

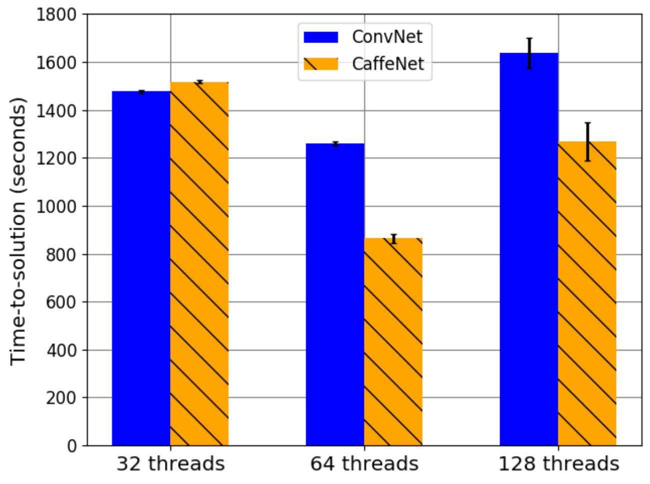

Figure 1: Scaleup Performance of ConvNet and CaffeNet with Varying Number of OpenMP Threads.

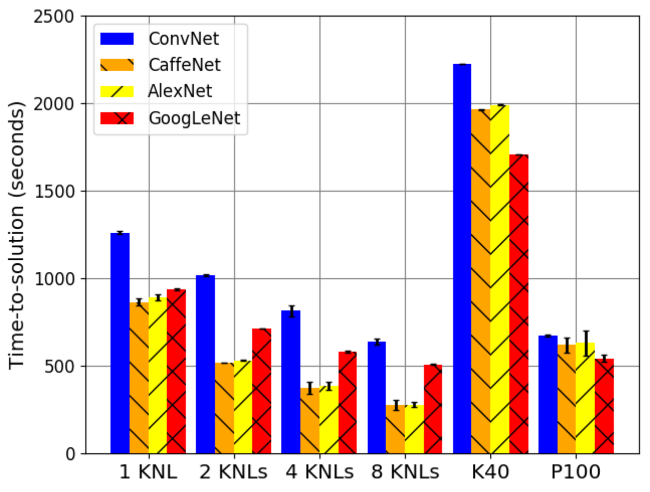

Figure 2: Strong Scaling Performance of ConvNet, CaffeNet, AlexNet, and GoogLeNet on Stampede2.

2.2 Strong Scaling of Intel-Caffe

Using 64 OpenMP threads with the MKL2017 engine delivers ~1.8-2.3x performance speedup for all four applications compared to that of one Nvidia K40 GPU. It is ~1.4-1.9x slower compared to that of one Nvidia P100 GPU, as shown in Figure 2. In the test cases of CaffeNet and AlexNet, two KNLs perform ~20% faster than a single P100 GPU, despite less than perfect scaling.

We then fix the problem size and measure the performance on varying scales. The scaling efficiency keeps dropping with increasing number of the nodes. ConvNet and GoogLeNet's scaling efficiencies are down to ~50% on eight KNLs, while the CaffeNet and AlexNet are at ~30% on eight KNLs.

We believe this is due to the unscalable implementation of the back-propagation algorithm, given the feed-forwarding process is embarrassingly parallel with the data parallel approach. And the gradient communication can be handled with MPI collectives in a scalable manner

2.3 Weak Scaling of Intel-Caffe

In this experiment, we seek to understand Intel-Caffe's weak scaling performance, as machine learning practitioners care about training and validation accuracy with a fixed number of epochs (an epoch is a traverse of all data items in the training datasets). Weak scaling can be useful if it can achieve high scaling efficiency.

We fix the batch size per node, so the total batch size is proportional to the node count. Figure 3 shows the weak scaling performance compared to that of one KNL. All four applications are able to keep up to ~80% scaling efficiency up to 512 KNLs. On 1,024 KNLs, the scaling efficiency drops significantly. This is due to an MLSL scalability issue, which we are investigating with Intel people. This issue blocks starting Intel-Caffe on 1,024 KNLs, and we have to turn off the MLSL server then run Intel-Caffe with pure MPI across nodes.

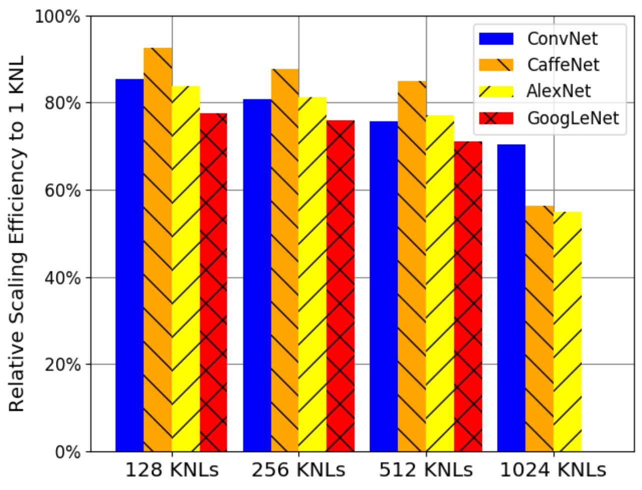

Figure 3: Weak Scaling Performance of ConvNet, CaffeNet, AlexNet, and GoogLeNet on Stampede2.

2.4 MXNet and TensorFlow

We also evaluate the performance of MXNet (v0.10.0-79-g790328f) and TensorFlow (v1.3.0-rc2) on one KNL node using a set of image classification application. In general, these applications are 1.2-3.7x and 5.0-22.3x slower than the performance on a Nvidia K40. This is due to the lack of KNL optimization in the source code. We notice that Intel has put effort into MXNet [16] and TensorFlow [19] optimization. We will keep the pace with Intel and provide the optimized software to our users at their earliest availability.

3 SUMMARY

Our goal is to enable and support scientists' exploration and exploitation of deep learning techniques in their research fields. So far, we have built three popular deep learning frameworks on the

Stampede2 supercomputer and profiled the single-node performance. Particularly for Intel-Caffe, we profiled and studied the strong-scaling performance and compared that to Nvidia's K40 and P100 GPU. We scale four image classification applications with real dataset up to 512 KNLs with ~80% efficiency in the weak scaling manner.

Technically, our next steps include scaling Intel-Caffe up to all 4,200 KNLs and scaling out MXNet and TensorFlow on the Stampede2 supercomputer and its Phase 2 Sky Lake processors. Strategically, we will release these deep learning frameworks and provide official support with training sessions. We will also invite and encourage users to try these deep learning frameworks on Stampede2 and possibly incubate a few research projects that leverage deep learning techniques.

REFERENCES

[1] CaffeNet. https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet.

[2] ConvNet. https://code.google.com/archive/p/cuda-convnet/.

[3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

[4] A. Aliper, S. Plis, A. Artemov, A. Ulloa, P. Mamoshina, and A. Zhavoronkov. Deep learning applications for predicting pharmacological properties of drugs and drug repurposing using transcriptomic data. *Molecular pharmaceutics*, 13(7):2524–2530, 2016.

[5] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5, 2014.

[6] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7, 2010.

[7] J. Carrasquilla and R. G. Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431–434, 2017.

[8] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[9] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew. Deep learning with cots hpc systems. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1337–1345, 2013.

[10] N. Codella, Q.-B. Nguyen, S. Pankanti, D. Gutman, B. Helba, A. Halpern, and J. R. Smith. Deep learning ensembles for melanoma recognition in dermoscopy images. *arXiv preprint arXiv:1610.04662*, 2016.

[11] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[13] Intel. Intel Distribution of Caffe. https://github.com/intel/caffe.

[14] Intel. Intel Machine Learning Scaling Library for Linux OS. https://github.com/01org/MLSL.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[16] Y. J. K. Installing and Building MXNet with IntelÂő MKL. https://software.intel.com/en-us/articles/installing-and-building-mxnet-with-intel-mkl.

[17] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[19] E. O. TensorFlow Optimizations on Modern Intel Architecture. https://software.intel.com/en-us/articles/tensorflow-optimizations-on-modern-intel-architecture.

[20] S. M. Plis, D. R. Hjelm, R. Salakhutdinov, and V. D. Calhoun. Deep learning for neuroimaging: a validation study. *arXiv preprint arXiv:1312.5847*, 2013.

[21] D. Stanzione, B. Barth, N. Gaffney, K. Gaither, C. Hempel, T. Minyard, S. Mehringer, E. Wernert, H. Tufo, D. Panda, and P. Teller. Stampede 2: The evolution of an xsede supercomputer. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*, PEARC17, pages 15:1–15:8, New York, NY, USA, 2017. ACM.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[23] D. Yu, A. Eversole, M. Seltzer, K. Yao, Z. Huang, B. Guenter, O. Kuchaiev, Y. Zhang, F. Seide, H. Wang, et al. An introduction to computational networks and the computational network toolkit. *Microsoft Technical Report MSR-TR-2014–112*, 2014.