Understanding I/O behavior of Scientific Deep Learning Applications in HPC systems

Hariharan Devarajan
Illinois Institute of Technology
hdevarajan@hawk.iit.edu

Huihuo Zheng
Argonne National Laboratory
huihuo.zheng@anl.gov

Xian-He Sun
Illinois Institute of Technology
sun@iit.edu

Venkatram Vishwanath
Argonne National Laboratory
venkatv@alcf.anl.gov

I. Extended Abstract

In the past decade, deep learning (DL) has been applied to a wide range of applications [1], [2], [3] to achieve unprecedented results. These include image recognition [4], natural language processing [5], and even autonomous driving [6], as well as physical science domains such as cosmology [7], materials science [8], [9], and biology [10], [11]. DL methods iteratively adjust the weights within the network to minimize a loss function. At each training step, the application reads a mini-batch of data, computes the gradient of the loss function and then updates the weights of the network using stochastic gradient descent. Many new AI hardware (e.g., GPU, TPU, Cerebras, etc.) have been designed and deployed to accelerate the computation during the training. However, as the size and complexity of the datasets grow rapidly, DL training becomes increasingly read intensive. I/O is a potential bottleneck in the DL applications [12]. On the other hand, more and more scientific DL studies are performed in high performance supercomputers through a distributed training framework to reduce the training time-to-solution [13]. Therefore, characterizing the I/O behavior of DL workloads in high-performance computing (HPC) environment is crucial for us to address any potential bottlenecks in I/O and to provide useful guidance in performing efficient parallel I/O.

In this study, we aim to understand the I/O behavior in scientific DL applications. As a starting point, we explore a collection of scientific deep learning workloads which are currently running at Argonne Leadership Computing Facility (ALCF). These workloads are selected from various projects, such as Argonne Data Science Program (ADSP), Aurora Early Science Program (ESP), and Exascale Computing Projects (ECP). The science domains represented by the workloads include neutrino physics [7], cosmology [14], materials science [8], computational physics [15], and biology [16], [10]. Many of the workloads are in active development targeting the upcoming future exascale supercomputers. One of the long term goals for this study is to identify any existing I/O bottlenecks in these workloads on current production machines and suggest I/O optimizations for current applications and as we develop these for future systems.

We profile the I/O behavior of eight DL applications on Theta, our current production leadership supercomputer at ALCF. To realize this, we utilize the profilers provided by the DL framework such as TensorFlow profiler as well as low-level I/O profiler such as Darshan, to study the I/O behavior of applications on supercomputers. These profilers are accompanied with their analysis tools. However, to get a holistic view of the application, we developed a Python library, VaniDL, for integrating and post-processing the information obtained from the profiling tools and generating high level I/O summary of the application. The main contributions of this work are:

1) proposing a systematic framework for I/O profiling for DL workloads and developing an analyzer tool, VaniDL, which provides insights on the I/O behavior of DL applications.
2) preliminary exploration of the I/O behavior of eight scientific DL applications on a leadership supercomputer.

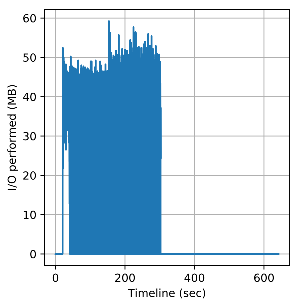
II. I/O Behavior of HPC Deep Leaning workloads

A. Methodology

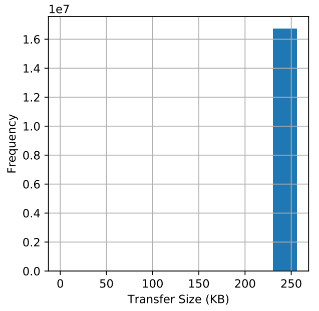
Applications: We target distributed DL workloads. These include Neutrino and Cosmic Tagging with UNet [7], Distributed Flood Filling Networks (FFN) for shape recognition in brain tissue [8], Deep Learning Climate Segmentation [17], CosmoFlow for learning universe at scale [14], Cancer Distributed Learning Environment (CANDLE) for cancer research [10], Fusion Recurrent Neural Net for representation learning in plasma science [16], Learning To Hamiltonian Monte Carlo (L2HMC) [15], and TensorFlow CNN Benchmarks [18]. These applications are implemented in TensorFlow and use Horovod for data parallel training. Some of them also have PyTorch implementation.

Hardware: We run the applications on Theta. This supercomputer consists of more than 3600 nodes and 864 Aries routers interconnected with a dragonfly network. Each router hosts four 2^{nd} generation Intel Xeon Phi^{TM} processors, coded name Knights Landing (KNL). Each node is equipped with 192 GB of DDR4 and 16 GB of MCDRAM. In all the studies presented here, we set 2 hyper-threads per core for a total of 128 threads per node, and four processes per node. The datasets are stored in Lustre file system. We set the Lustre stripe size to be 1 MB and stripe count to be 48.

Tools: We use Darshan (with extended tracing) as our low-level I/O profiling tool along with TensorFlow profiler. Additionally, we process the profiling results using our custom analytic tool, VaniDL [19] to integrate the low-level Darshan logs with high-level TensorFlow profiler logs and generate a



(a) I/O timeline.



(b) Transfer size distribution for all the I/O operations.

Fig. 1. CosmoFlow I/O Behaviors: sub-figure a) showcases that 8% of the application time is spent on I/O. sub-figure b) showcases the distribution of the transfer size for all the I/O operations. It shows that all the I/O requests has transfer size of 256 KB which is the default setting in TFRecord APIs.

holistic I/O information of the application, such as I/O access pattern, transfer size distributions for all the I/O operations, I/O access timeline, etc. All the results described below are the outcome of the analysis produced by VaniDL.

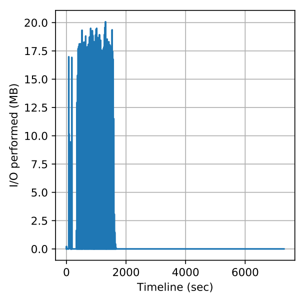
B. I/O analysis of Scientific Deep Learning Applications

In this section, we present the characteristics of 2 (of the 8) DL applications. The applications were run on 128 nodes with 4 processes per node (total of 512 processes).

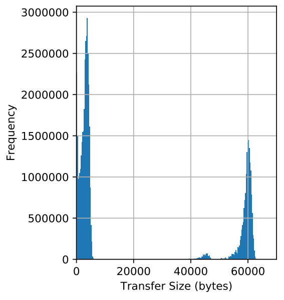
1) CosmoFlow [14]: CosmoFlow is a 3D convolutional neural network model for studying the features in the distribution of dark matter. The dataset contains 327,680 images, each of size (128, 128, 128, 4). They are stored in 1,024 TFRecord files, each of which contains 256 samples. The total size of the dataset is 2 TB. The application uses TensorFlow data pipeline to stream the data from the file system. The whole process includes randomly selecting a subset of files to read for each rank and pre-processing the samples. In the benchmark, we set the batch size to be one, and ran the application for 256,000 steps (4 epochs).

As shown in Figure 1, the benchmark was executed for 624 seconds, out of which 8% is spent in I/O. The whole dataset (2 TB) was read entirely in parallel during the first epoch and cached in the memory as the dataset fits into the total memory of 128 nodes on Theta (shown in Figure 1(a) from VaniDL). For the later epochs, the data were read directly from the cache instead of going to the file system. Therefore, there is no I/O performed for the later epochs. The transfer size (the size of the read buffer) for each read operation is 256 KB. The I/O is consecutive with an aggregate read bandwidth of 145 GB/s.

2) Distributed Flood Filling Networks [8]: It is a recurrent 3D convolutional network for segmenting complex and large shapes of neurons from a raw image of brain tissue. The dataset is stored in two HDF5 files, one for real data and the other for metadata associated with the dataset (e.g., size of samples, location of samples within the dataset, etc.). The dataset is composed of 4,096 grayscale map images of size 32×32×32. The total size of the dataset is 2.28 GB. One thing to mention is that the application uses a 4K field of view to traverse each sample. Therefore, only part of each sample within the field of view is accessed.



(a) I/O Timeline



(b) Transfer size distribution for all the I/O operations.

Fig. 2. FFN I/O Behaviors: sub-figure a) showcases the consecutive I/O on the HDF5. After the first epoch the data is accessed from the HDF5 chunk cache. sub-figure b) showcases how small access (i.e., 100 B) happens on the metadata file and 60 KB happens on the images maps file.

As shown in Figure 2, the application spent 1,314 seconds in I/O, which is 18.4% of the overall execution time. The I/O does not overlap with computations. The I/O happens only at the first epoch since the data was directly fetched from the chunk cache for the subsequent epochs. At each training step, each process randomly reads an image from the data file based on the location provided from the metadata. The distribution of the transfer size centers at two places [Figure 2(b)], one at about 62 KB (for reading images), the other at about 100B (for reading metadata). A chunk size of 65KB is used in the data file. That is why the transfer size for reading images is about 62KB. Because the transfer size is relatively small and the access pattern is random due to shuffling, the aggregate read bandwidth for the application is only about 42 GB/s. One can potentially read a larger buffer of images and perform in-memory shuffling to improve the I/O bandwidth.

In conclusion, we have analyzed the I/O in selected scientific workloads at current stage of development. Currently, the datasets are still relatively small and fit into the aggregate memory of the compute nodes. I/O only occurs at the first epoch. However, as the datasets grow larger than the main memory, I/O will happen at every epoch and might potentially become a bottleneck. We observe that the TensorFlow Data Pipeline enables efficient I/O by overlapping the I/O with computation in a pipepline fashion. However, it does not support parallel I/O, which will potentially limit its I/O scalability. Additionally, we observe that many scientific applications utilizing data formats such as HDF5 and NPZ use their custom I/O functions instead of framework provided data APIs can be further optimized, particularly in the area of overlapping I/O with computation through data prefetching and in-memory shuffling to avoid any performance degradation due to random I/O. These are area of research we plan to explore in future.

ACKNOWLEDGMENT

We would like to thank the application developers including Sam Foreman, Rafael Vescovi, Corey Adams, Murali Emani, Kyle Gerard Felker, Xingfu Wu, and Murat Keceli for providing the workloads and helpful discussion in running the workloads on Theta. This work used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility under Contract DE-AC02-06CH11357.

References

[1] Z. Jiang, W. Gao, L. Wang, X. Xiong, Y. Zhang, X. Wen, C. Luo, H. Ye, X. Lu, Y. Zhang *et al.*, "Hpc ai500: a benchmark suite for hpc ai systems," in *International Symposium on Benchmarking, Measuring and Optimization*. Springer, 2018, pp. 10–22.

[2] S. Shams, R. Platania, K. Lee, and S.-J. Park, "Evaluation of deep learning frameworks over different hpc architectures," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1389–1396.

[3] C. Chin and D. E. Brown, "Learning in science: A comparison of deep and surface approaches," *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching*, vol. 37, no. 2, pp. 109–138, 2000.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[6] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[7] C. J. Adams, "Neutrino and Cosmic Tagging with UNet," 2015. [Online]. Available: https://github.com/coreyjadams/CosmicTagger

[8] W. Dong, M. Keceli, R. Vescovi, H. Li, C. Adams, E. Jennings, S. Flender, T. Uram, V. Vishwanath, N. Ferrier *et al.*, "Scaling distributed training of flood-filling networks on hpc infrastructure for brain mapping," in *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*. IEEE, 2019, pp. 52–61.

[9] G. B. Goh, N. O. Hodas, and A. Vishnu, "Deep learning for computational chemistry," *Journal of computational chemistry*, vol. 38, no. 16, pp. 1291–1307, 2017.

[10] X. Wu, V. Taylor, J. M. Wozniak, R. Stevens, T. Brettin, and F. Xia, "Performance, energy, and scalability analysis and improvement of parallel cancer deep learning candle benchmarks," in *Proceedings of the 48th International Conference on Parallel Processing*, 2019, pp. 1–11.

[11] O. Faust, Y. Hagiwara, T. J. Hong, O. S. Lih, and U. R. Acharya, "Deep learning for healthcare applications based on physiological signals: A review," *Computer methods and programs in biomedicine*, vol. 161, pp. 1–13, 2018.

[12] S. W. Chien, S. Markidis, C. P. Sishtla, L. Santos, P. Herman, S. Narasimhamurthy, and E. Laure, "Characterizing deep-learning i/o workloads in tensorflow," in *2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS)*. IEEE, 2018, pp. 54–63.

[13] Y. Kwon and M. Rhu, "Beyond the memory wall: A case for memory-centric hpc system for deep learning," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018, pp. 148–161.

[14] A. Mathuriya, D. Bard, P. Mendygral, L. Meadows, J. Arnemann, L. Shao, S. He, T. Kärnä, D. Moise, S. J. Pennycook *et al.*, "Cosmoflow: Using deep learning to learn the universe at scale," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 819–829.

[15] D. Levy, M. D. Hoffman, and J. Sohl-Dickstein, "Generalizing hamiltonian monte carlo with neural networks," *arXiv preprint arXiv:1711.09268*, 2017.

[16] G. Dong, K. G. Felker, A. Svyatkovskiy, W. Tang, and J. Kates-Harbeck, "Fully convolutional spatio-temporal models for representation learning in plasma science," *arXiv preprint arXiv:2007.10468*, 2020.

[17] T. Kurth, S. Treichler, J. Romero, M. Mudigonda, N. Luehr, E. Phillips, A. Mahesh, M. Matheson, J. Deslippe, M. Fatica *et al.*, "Exascale deep learning for climate analytics," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 649–660.

[18] Tensorflow, "TensorFlow benchmarks," 2018. [Online]. Available: https://github.com/tensorflow/benchmarks

[19] H. Devarajan, "VaniDL: Deep Learning I/O Analyzer," 2020. [Online]. Available: https://github.com/hariharan-devarajan/vanidl