Toward Training a Large 3D Cosmological CNN with Hybrid Parallelization

Yosuke Oyama[1,2], Naoya Maruyama[2], Nikoli Dryden[3,2], Peter Harrington[4], Jan Balewski[4], Satoshi Matsuoka[5,1], Marc Snir[3], Peter Nugent[4], and Brian Van Essen[2]
oyama.y.aa@m.titech.ac.jp
[1] Tokyo Institute of Technology
[2] Lawrence Livermore National Laboratory
[3] University of Illinois at Urbana-Champaign
[4] Lawrence Berkeley National Laboratory
[5] Riken Center for Computational Science

1 INTRODUCTION

Training of Convolutional Neural Networks (CNNs) has been drastically accelerated by exploiting mini-batch Stochastic Gradient Descent (SGD) in the last decade. "Data-parallel" training in the context of mini-batch SGD means that each processor, such as a CPU or a GPU, 1) holds the same copy of the network parameters and 2) computes their gradients with respect to a specific disjoined subset of a mini-batch, and 3) performs collective aggregation to update the parameters. Since it only requires coarse-grained inter-processor communication and has near-perfect load balance, data-parallel training has been exploited to train many types of CNNs.

However, in extreme-scale training, model-parallelism is also required for various reasons. In model-parallel (or "hybrid-parallel" where both strategies are used at the same time) training, a single independent model is split into multiple processors, incurring fine-grained communication to exchange activations and errors in the middle of the network. One of the important advantages of model-parallelism over data-parallelism is that it relives memory requirements for each processor. Hence it enables to train a bigger model than what can be trained under the data-parallel scheme.

In this paper, we demonstrate that the CosmoFlow network [4], a 3D CNN to predict cosmological parameters from 3D mass distribution, is scaled to 128 nodes and 64 larger example size utilizing hybrid-parallelism.

1.1 CosmoFlow

CosmoFlow [4] is a project to estimate the values of important cosmological parameters from 3-dimensional universe data by using deep learning. In the previous work, the authors first conduct thousands of independent N-body simulations with varied initial cosmological parameters, and then construct a dataset to predict the parameters from simulated universes.

In this work, we use the "4perE" dataset, composed of 1,027 data samples each of which is $4 \times 512 \times 512 \times 512$ voxels, where 4 is the number of channels (C), and 4 cosmological parameters. We synthesize two datasets of 4×128^3 and 4×256^3 voxels, by splitting each 4×512^3 voxels from the original dataset with the same size.

2 IMPLEMENTATION DETAILS

2.1 Distconv

Distconv [1] is a hybrid-parallel kernel library for CNNs, primarily designed for Livermore Big Artificial Neural Network Toolkit (LBANN) [5]. Distconv applies one of the model-parallel strategies, spatial partitioning, to convolutional and pooling layers. The basic concept of this partitioning follows parallelized stencil computations. First, convolution is performed to the center part of an input tensor, and at the same time a halo exchange is started in a different asynchronous CUDA stream among GPUs in the same sample group. We repeat the one-dimensional halo exchange three times to perform the three-dimensional halo exchange. Once halo exchange is completed, convolution is performed on the halo region in the stream. Distconv exploits the Aluminum [2] GPU-aware asynchronous communication library to perform efficient halo exchanges.

2.2 I/O performance optimization

We study the performance of two different data sample readers: The "Direct" data reader loads data samples from the file system directly, or from a node-local SSD where the entire dataset is preloaded in advance of training. On the other hand, the "Conduit" data reader uses Conduit [3] as an I/O backend. Conduit is an open source data exchange library that provides efficient ways of exchanging scientific data between applications, exchanging data between different processes within a single MPI based application, and managing in-memory data movement within a single process. Our Conduit data reader preloads the entire dataset from the file system into CPU memory before training starts. The process that performs the read thereafter "owns" the data. Subsequently, prior to each minibatch, we employ an MPI-based data exchange to shuffle the data to the process that requires it. After a mini-batch is loaded, the MPI processes perform data shuffle operation to exchange the desired spatial parts of the mini-batch.

2.3 Network

We use a CNN composed of seven 3D convolutional layers and three fully-connected layers derived from the original work. The per-sample memory requirements for the input width of $W = 128$ and $W = 256$ are under memory size of the latest GPUs (0.82 GiB and 6.59 GiB respectively). For instance, an NVIDIA Tesla V100 GPU has 16 GB memory which is capable of holding one or more samples of the datasets. Thus, data-parallel training is the most efficient way to train the networks as it only requires a global collective communication in an iteration. For $W = 512$, however, the memory requirement exceeds the memory size (52.7 GiB). Hence it is not feasible to perform data-parallel training.

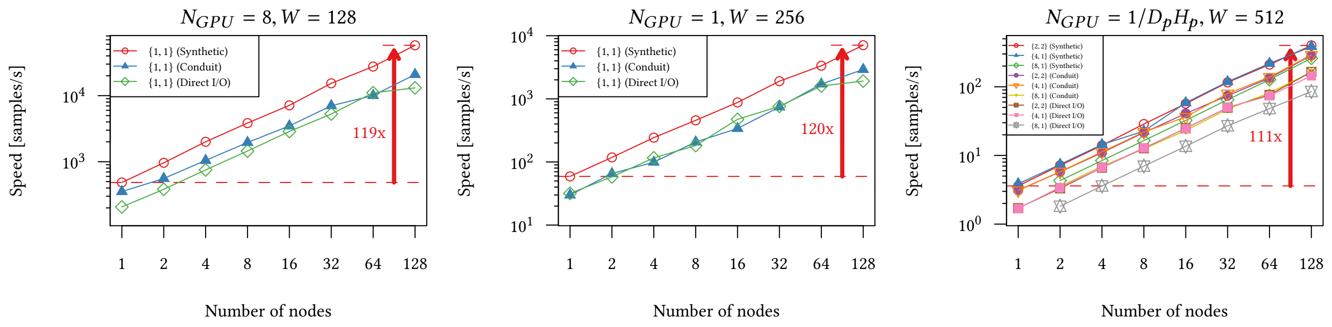

Figure 1: Weak scaling of the CosmoFlow network. N_{GPU} represents the number of data samples per GPU, and $\{D_p, H_p\}$ represents the depth and the height dimensions are distributed among D_p and H_p process groups respectively.

3 EVALUATION

Figure 1 shows the mini-batch weak scaling of our implementation with three different input widths. We use per-GPU batch sizes of 8, 1, 1/4 (1 for each node) and 1/8 (1 for each node) for $W = 128, 256, 512$ (with $\{D_p, H_p\} = \{4, 1\}$ and $\{D_p, H_p\} = \{2, 2\}$ configurations, where D_p, H_p denote the number of partitions for the depth and the height dimensions respectively) and $W = 512$ (the $\{D_p, H_p\} = \{8, 1\}$ configuration) datasets respectively. We run the framework for few epochs, and show the minimum iteration time of the last epoch. In this experiment, we also measure the performance with a "Synthetic" dataset configuration, where the I/O process is skipped so that the pure computation and communication performance is measured. We use Lassen, a GPU cluster of Lawrence Livermore National Laboratory, each node of which equips four NVIDIA Tesla V100 GPUs. Each node has two IBM Power9 CPU chips with 256 GB memory and four NVIDIA Tesla V100 GPUs with 16 GB memory and NVLink links. Lassen adopts 6 NVLink links between GPU-GPU and GPU-CPU with 300 GB/s total bandwidth, and 100 Gb/s EDR InfiniBand among computing nodes.

In all of the cases, our implementation achieves nearly linear speedup up to 128 compute nodes. We achieve a speedup of 119x and 120x on 128 nodes over 1 node with the $W = 128$ and $W = 256$ datasets respectively. When $W = 512$, however, it is infeasible to perform data-parallel training since the model is too huge to fit into GPU memory as mentioned. With our implementation, however, we achieve 111x of speedup over 1 node by exploiting hybrid-parallelism even if layer-wise communication is introduced. In the experiment with $W = 512$, we use the minimum number of nodes for the batch size of one, and then increase the number of nodes in weak scaling fashion. Thus, on 128 nodes, we use a mini-batch size of 128 for $\{4, 1\}$ and $\{2, 2\}$, but 64 for $\{8, 1\}$. Even though this configuration degrades per-sample computation efficiency for $\{8, 1\}$, it also introduces the possibility of parallelizing the computation on each data sample among more GPUs. Indeed, when the mini-batch size is 64, the computation speed with $\{4, 1\}$ (on 64 nodes) is 218.3 samples/s, while it is 260.0 samples/s with $\{8, 1\}$ (1.19x of $\{4, 1\}$), even if inter-node layer-wise communication is required. It achieves 1.42 PFlop/s on 128 nodes with the $\{2, 2\}$ configuration and the synthetic data reader with $W = 512$, and 289 TFlop/s without the synthetic data reader.

4 CONCLUSIONS

In this work, we demonstrated that our framework successfully accelerates training of the CosmoFlow network by introducing hybrid-parallelism, achieving 1.42 PFlop/s on 128 nodes (512 Tesla V100 GPUs). To best of our knowledge, our work is the first attempt to train a 3D CNN with the CosmoFlow dataset whose input size is 4×512^3 voxels without partitioning the data samples. Our experimental results showed that the performance possibility of hybrid-parallelism for 3D convolutional neural networks.

ACKNOWLEDGMENT

This research was supported by JSPS KAKENHI Grant Number JP18J22858, Japan. This research was supported by the Exascale Computing Project (17-SC-20-SC), This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231. LLNL-ABS-776502.

REFERENCES

[1] Nikoli Dryden, Naoya Maruyama, Tom Benson, Tim Moon, Marc Snir, and Brian Van Essen. 2019. Improving Strong-Scaling of CNN Training by Exploiting Finer-Grained Parallelism. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS '19).*

[2] Nikoli Dryden, Naoya Maruyama, Tim Moon, Tom Benson, Andy Yoo, Marc Snir, and Brian Van Essen. 2018. Aluminum: An Asynchronous, GPU-Aware Communication Library Optimized for Large-Scale Training of Deep Neural Networks on HPC Systems. In *2018 IEEE/ACM Machine Learning in HPC Environments (MLHPC).* 1–13. https://doi.org/10.1109/MLHPC.2018.8638639

[3] Lawrence Livermore National Laboratory. 2019. Conduit. https://github.com/LLNL/conduit

[4] Amrita Mathuriya, Deborah Bard, Peter Mendygral, Lawrence Meadows, James Arnemann, Lei Shao, Siyu He, Tuomas Kärnä, Diana Moise, Simon J. Pennycook, Kristyn Maschhoff, Jason Sewall, Nalini Kumar, Shirley Ho, Michael F. Ringenburg, Prabhat, and Victor Lee. 2018. CosmoFlow: Using Deep Learning to Learn the Universe at Scale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '18).* IEEE Press, Piscataway, NJ, USA, Article 65, 11 pages. http://dl.acm.org/citation.cfm?id=3291656.3291743

[5] Brian Van Essen, Hyojin Kim, Roger Pearce, Kofi Boakye, and Barry Chen. 2015. LBANN: Livermore Big Artificial Neural Network HPC Toolkit. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments (MLHPC '15).* ACM, New York, NY, USA, Article 5, 6 pages. https://doi.org/10.1145/2834892.2834897