研 究 終 了 報 告 書

「分散型ディープニューラルネットワークの大規模設計の調査・研究」

研究期間： 2019 年 10 月～2022 年 03 月
研 究 者： ThaoNguyen Truong

1． 研究のねらい

The community puts a significant effort to speed up the Deep Learning (DL) design process, e.g., training large Deep Neural Networks (DNNs) on large-scale High-Performance Computing (HPC) systems. With the steady increase in the size of datasets and DNNs model, the conventional data parallelism approach faces substantial scalability challenges includes (a) lack of memory capacity at computing nodes, (b) high I/O and communication overhead, and (c) application accuracy. The target of this project is to enable training Deep Learning models on large-scale HPC systems with a short time by studying (1) new parallelism strategies (not only data parallelism), e.g., model/hybrid parallelism, (2) methods to reduce the I/O and communication time. Based on this research, we aim to develop frameworks that support DL developers and researchers in figuring out the optimal large-scale strategies when deploying a given DNNs model, dataset, and computing system.

2． 研究成果

（1）概要

We conduct the following research topic:

Research theme 1: We firstly investigated the methods that help to resolve the problem of lacking memory capacity when training an ultra-big DL model by investigating the model/hybrid parallelism. We developed a performance/memory estimation model as the basis for a utility named ParaDL, that aids end-users, framework developers, and system builders in identifying the optimal large-scale parallelism strategies (published in conference-HPDC2021). The proposed estimation model is helpful in solving different emergent scalability problem of large-scale training Deep Neural Network.

Research theme 2: We designed the methods that help to reduce up to 60% of communication time for both data and model parallelism. Basically, state-of-the-art techniques for optimizing communication time are (i) Architecture-aware algorithm, (ii) data compression, and (iii) comm. /comp. overlapping. We proposed the communication algorithm optimized for GPU-cluster system (published in conference-CCGRID2021), We also study how to combine those three techniques.

Research theme 3: We found that loading data samples from dataset to computing devices (I/O time) during training is one of the main time-consuming tasks. Especially when the dataset size is too big for impossibly replicating the entire dataset into the local storage of computing devices. We investigate the viability of partitioning the dataset among computing devices and only doing a partial distributed exchange of samples before each epoch. We provide a general solution, implemented in PyTorch, that allows the user to control the partitioning and partially

distributed exchange scheme. This would enable users to experiment with large datasets to explore the viability of our scheme for their specific dataset/model/scale (conditionally accepted in IPDPS2022).

（2）詳細

Research Theme 1: Performance/Memory Estimation Model (collaborated with Barcelona Supercomputing Center (BSC):

We anticipate that model/hybrid parallelism approaches would have a central role in the scaling of DL training, especially for scientific simulations that more than often deal with high-resolution datasets. We addressed (all) different types of parallelism approaches based on parallelizing different dimension in training Deep Neural Network (DNN) / Deep Learning (DL) application including: (1) the size of data set (data-parallelism or sample parallelism), (2) dimension of each data sample such as width, height of an image sample (spatial parallelism), (3) depth of DL model, i.e., the number of layers (vertical model parallelism or layer parallelism), (4) the width of DL model, i.e., the channel /filters of each layer (horizontal model parallelism), and the combination for these above approach (hybrid parallelism). we investigate the computation, communication, and memory requirements to understand the effect of parallelism choices on performance. It is worth to noting that the analytical model predicts the ideal performance without considering (i) the effects of both self-contention (cause by multiple flows use some links at the same time) and network congestion (caused by other applications running at the same times on a shared system), (ii) the effects interconnect hierarchy of modern computing systems, the algorithms used by communication libraries, and the communication technologies to the input-parameter of our analytical model.

We rely on a **hybrid of analytical modeling and empirical parameterization**. We introduce the self-contention modeling with a contention penalty coefficient parameter which can be estimated analytically by using dynamic contention graphs. In addition, the baseline performance predicted by our analytical model can be complemented with a congestion impact factor, which can be empirically estimated for each specific computing system beforehand. For issue (ii), we use the empirical parameterization method, i.e, we empirically measure the communication time of collective communication patterns, such as Allreduce, with different message sizes, number of involved processing elements on a specific computing system. Those empirical measurements can be derived from well-known tools for the performance of systems, e.g., OSU Micro-Benchmarks or NCCL-test. Figure 2 show an example of the prediction accuracy of our new estimation model which can achieve up to 97% correctness, and 86.7% on average across all parallel strategies on multiple CNN models and datasets on up to 1K GPUs (more result in paper [1], HPDC2021).

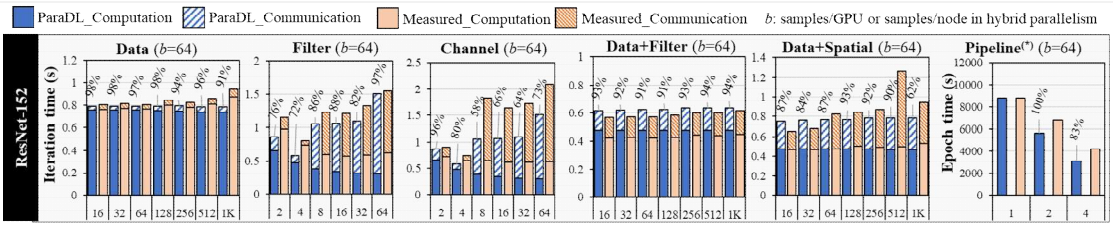
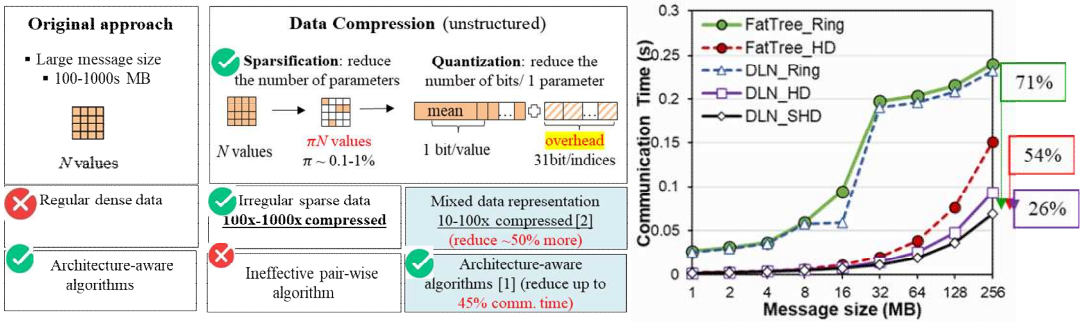


Figure 2: Time breakdown of our analytical model (ParaDL) in comparison with measured runs for different parallelism strategies of DL training. The x axis is the number of GPUs. The label above each column shows the projection accuracy.

Research Theme 2: Communication Optimization – study the method to reduce the communication time.

Communication is one of the factors that limits the scaling of training deep learning across multiple compute nodes of the HPC system. At each training iteration, each computing process performs the Allreduce collective communication to share their local training result, e.g., local gradients to calculate the global gradients for weight update. As the deep learning models become more complex due to accuracy demands, communication becomes a critical bottleneck especially as the message sizes become bigger. To cope with this problem, one approach is to optimize the Allreduce algorithm for large messages by considering the network architecture (topology-aware). The other is to reduce the amount of transferred data using data compression methods. In the latter approach, sparsification is a promising method that helps to cut down the transferred data significantly, e.g., up to 99.9%, without losing validation accuracy. However, the practical use of sparsification is limited because the sparse data restricts the communication into the ineffective pair-wise scheme. In this study, we propose an Allreduce algorithm that exploits both topology-aware techniques CCPE journal, I/F 1.167) and data compression (IPCCC2019) optimized for AI-accelerated computing systems such as ABCI and NVCluster. We validate our algorithmic implementation on Simgrid simulation. The results show significant improvement, as in Figure 3a, of our proposed over the conventional Allreduce and to the sparse Allreduce with a trivial extra computing cost, e.g., 2-3% of total time.



(a) Overview of our proposal (b) Example result with 512GPUs

Figure 3: Overview of our proposal and example result in research theme 2

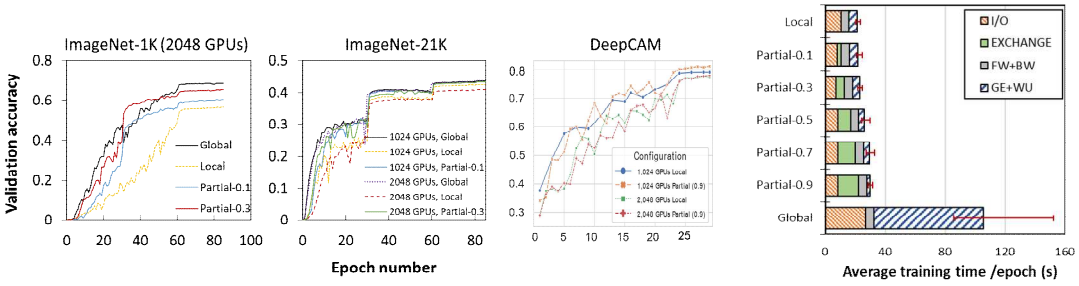
In [2] (published in CCGRID2021), we also investigate the benefits of co-design of Allreduce algorithms and the network system to reduce the communication time of training

process. We propose to replace the Fat-tree network topology with a variant of Distributed Loop Network topology (DLN) that guarantees a fixed routing paths length between any pairs of computing nodes for the communication pattern of halving-doubling Allreduce algorithm.We also propose a technique to eliminate/mitigate the network contention by using our performance estimation model (SHD). The simulation results show that our co-design helps to reduce the communication time significantly (50-90%) when the number of ranks increase up to 1000s GPUs (as shown in Figure 3b).

Research Theme 3: I/O Optimization – study the method to reduce the I/O time (collaborated with RIKEN:

Distributing the training of a neural network in a data (and hybird) parallelism fashion over compute nodes of a supercomputer requires loading the input samples on compute nodes so that each node can process a subset of the samples at each training epoch. This is either done by storing the entire dataset on computing node-local storage, or by each node reading a subset of the samples from the parallel file system (PFS). As datasets become larger, storing the entire dataset on local storage becomes impossible since they exceed storage capacities. Similarly, reading from the parallel files system puts enormous pressure on the storage nodes because many compute nodes read terabytes of data simultaneously. Moreover, to improve generalization, distributed neural network training shuffles the data at each epoch so that nodes can randomly access input samples, which further increases the I/O requirements of deep learning applications.

In [3] (IPDPS2021), we revisit data shuffling strategies when scaling deep learning applications to a large number of computing nodes. We develop a method that allows us to control the fraction of the dataset that is globally shuffled in each epoch. In our method we implement the following shuffling strategies. Global shuffling (i.e., all of the dataset is shuffled and distributed across workers), local shuffling (i.e., each worker uses the same part of the dataset for each epoch), and a novel partial-local shuffling strategy that exchanges only a configurable proportion of the dataset among workers in each epoch and leaves the rest local. Using our method requires very few changes to existing PyTorch training scripts, does not require any changes to the PyTorch framework itself, and could support any arbitrary format for the samples by implementing a loader for the data.



(a) Training accuracy at large scale (b) Training time (512GPUs)

Figure 4: Example result in research theme 3. Our proposed Partial Shuffling has accuracy as

good as the conventional global shuffling while achieving training time as fast as local shuffling.

We provide a proof that partial local shuffling produces the same gradients as global shuffling. We further investigate the practical conditions for assuring that the shuffling error without dominating the convergence rate. For empirical results, we use our solution to train DNNs on several datasets and we study the impact of various shuffling strategies on accuracy when scaling to a large number of computing nodes. The partial-local shuffling strategy then achieves similar accuracy as global shuffling while only requiring to store to up to 0.03% of the whole dataset (as shown in Figure 4). In addition, for data sets that do not fit locally in the first place, partial-local shuffling can improve accuracy compared to the local only access. This opens the doors for leveraging the potential of locality in large scale training of large datasets, and addresses the DL I/O challenge at its root: avoid I/O when possible.

3. 今後の展開

Because we provide frameworks that are designed for the general training tasks in this project such as ParaDL, KARMA, and PartialShuffling method in Pytorch, it is easy to use and apply our proposal for training different DNN models, datasets, and systems (not just our experiment result). For communication optimization, we implemented the new Allreduce algorithm with the basic Message Passing Interface in C, which could be run in both the simulation environment as well as the real computing system. Integrating those techniques in the future becomes possible and does not require large efforts.

In the near future, our research is still helpful because the trends of designing a bigger Deep Learning model still not ended. For example, the Natural Languages Processing (NLP) models using Transformer techniques, which is paid attention by researchers and big companies in recent 2 years increased their sizes frequently which could become 100s Billions of parameters. In this context, we will continue design a method to search the best (fine-grained) hybrid parallelism approach for a specific application such as NLP, by using our prediction model as the objective function of the optimization problems.

4. 自己評価

Our research helps to speed up the development of new Deep Learning applications. It could help to enlarge the usage of Deep Learning (DL), especially, DNN models in many different application areas such as robotics, medical science, music, as well as scientific research etc. where the datasets, DL models is much bigger than those are in the conventional computer vision area. In 5-10 years, our research could be helpful in studying the training Deep Learning application in the edge-computing environments. As the number of mobile devices increases, as the network infrastructure becomes faster, there is a trend that decentralized training the Deep Learning model at the edge mobile devices instead of centralized training at the high-performance computing system. It helps to resolve the problem of privacy where user's data do not leak to any company server for training. However, mobile devices could not have a high

configuration in terms of computing, memory, and energy resource. Our research on solving memory capacity issues and communication optimization such as compression/sparsification could help in this situation. Besides, the data in each edge mobile device is non-independent and identical, our theoretical research on shuffling the data could help in this case.

５．　主な研究成果リスト

（1）代表的な論文（原著論文）発表

研究期間累積件数：9 件

1. Truong Thao Nguyen, Albert Kahira, Leonardo Bautista Gomez, Mohamed Wahib, Ryousei Takano, Rosa Badia, “An Oracle for Characterizing and Guiding Large-Scale Training of Deep Neural Networks,” ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC2021), 2021,
Deep Neural Network (DNN) frameworks use distributed training to enable faster time to convergence and alleviate memory capacity limitations when training large models and/or using high dimension inputs. With the steady increase in datasets and model sizes, model/hybrid parallelism is deemed to have an important role in the future of distributed training of DNNs. We analyze the compute, communication, and memory requirements of Convolutional Neural Networks (CNNs) to understand the trade-offs of different parallelism approaches on performance and scalability.We leverage our model-driven analysis to be the basis for an oracle utility which can help in detecting the limitations and bottlenecks of different parallelism approaches at scale. We evaluate the oracle on six parallelization strategies, with four CNN models and multiple datasets (2D and 3D), on up to 1024 GPUs. The results demonstrate that the oracle has an average accuracy of about 86.74% when compared to empirical results, and as high as 97.57% for data parallelism.
2.　Nguyen, Truong Thao, and Mohamed Wahib. "An Allreduce Algorithm and Network Co-design for Large-Scale Training of Distributed Deep Learning." 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid). IEEE, 2021
Distributed training of Deep Neural Networks (DNNs) on High-Performance Computing (HPC) systems is becoming increasingly common. HPC systems dedicated entirely or mainly to Deep Learning (DL) workloads are becoming a reality. The collective communication overhead for calculating the average of weight gradients, e.g., an Allreduce operations, is one of the main factors limiting the scaling of data parallelism training. Several active efforts across different layers of the training stack including the training algorithms, parallelism strategy, communication algorithms, and system design have been proposed to cope with this communication challenge when scaling distributed training of DNNs. However, even with those methods, communication still becomes a bottleneck with the steady increase in model sizes, e.g.,100-10,000s MB, and the number of compute nodes, e.g., 1,000-10,000s of GPUs. In this work, we investigate the benefits of co-design of Allreduce algorithms and the network system. We propose to replace the Fat-tree network topology with a variant of Distributed Loop Network topology that guarantees a

fixed routing paths length between any pairs of computing nodes for the communication pattern of halving-doubling Allreduce algorithm.We also propose a technique to eliminate/mitigate the network contention.
3. Truong Thao Nguyen, François Trahay, Jens Domke, Aleksandr Drozd, Emil Vatai, Jianwei Liao, Mohamed Wahib, and Balazs Gerofi, "Why Globally Re-shuffle? An I/O Perspective on Data Shuffling in Large Scale Deep Learning", 36th IEEE International Parallel & Distributed Processing Symposium (IPDPS2022), 2022 (**conditional accepted**)
Replicating the entire dataset on the local storage is not feasible, for most HPC systems. In this paper we investigate the viability of partitioning the dataset among workers and only doing a partial distributed exchange of samples before each epoch, when deemed necessary. We show that partitioning and exchange schemes yield a shuffling error that, in theory, dominates the convergence rate. We show, however, through an extensive set of experiments that in practice the validation accuracy could be maintained when carefully tuning the partial distributed exchange. We provide a general solution, implemented in PyTorch, that allows the user to control the partitioning and partial distributed exchange scheme. This would enable users experimenting with large datasets to explore the viability of our scheme for their specific dataset/model/scale.

（2）特許出願

研究期間全出願件数：0 件（特許公開前のものも含む）

（3）その他の成果（主要な学会発表、受賞、著作物、プレスリリース等）

Not available