An Oracle for Guiding Large-Scale Model/Hybrid Parallel Training of Convolutional Neural Networks

Albert Njoroge Kahira* ,Truong Thao Nguyen†, Leonardo Bautista Gomez*, Ryousei Takano †, Rosa M Badia*, Mohamed Wahib†

*Barcelona Supercomputing Center, Barcelona, Spain
†National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

ABSTRACT

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 between 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 .

KEYWORDS: Deep Learning, Model Parallelism, Performance Modeling

1 Introduction

DNNs are achieving outstanding results in a wide range of applications, including image recognition, video analysis, natural language processing [S+14], and drug discovery [W+15], among many others. In the quest to increase solution accuracy, researchers are increasingly using larger training datasets as well as larger and deeper DNN models [B+18, Y+18, H+18]. In addition, applying Deep Learning (DL) in new domains, such as health care and scientific simulations, introduce larger data samples and more complex DNN models [K+18]. Those trends make the DNN training computationally expensive for a single node. Therefore, large-scale parallel training on high-performance computing (HPC) systems or clusters of GPUs is becoming increasingly common to achieve faster training times for larger models and datasets [B+18].

[1]E-mail: {nguyen.truong,takano-ryousei, mohamed.attia}@aist.go.jp, {albert.kahira,leonardo.bautista, rosa.m.badia}@bsc.es
[2]The first and second author contributed equally to this paper.

Table 1: Computation, Communication, and Memory Analysis Summary (per epoch)

	Computation Time T_{comp}	**Communication Time** T_{comm}	**Maximum Memory Per PE**	**Number of PEs** p
Serial	$D\sum_{l=1}^{G}\Big(FW_l + BW_l\Big) + \frac{D}{B}\sum_{l=1}^{G}(WU_l)$	0	$\gamma\delta\sum_{l=1}^{G}\Big(2B(\lvert x_l\rvert + \lvert y_l\rvert) + 2\lvert w_l\rvert + \lvert bi_l\rvert\Big)$	$p = 1$
Data	$\frac{D}{p}\sum_{l=1}^{G}(FW_l + BW_l) + \frac{D}{B}\sum_{l=1}^{G}(WU_l)$	$2\frac{D}{B}(p-1)\Big(\alpha + \frac{\sum_{l=1}^{G}\lvert w_l\rvert}{p}\delta\beta\Big)$	$\gamma\delta\sum_{l=1}^{G}\Big(\frac{2B}{p}(\lvert x_l\rvert + \lvert y_l\rvert) + 2\lvert w_l\rvert + \lvert bi_l\rvert\Big)$	$p \leq B$
Spatial	$\frac{D}{p}\sum_{l=1}^{G}\Big(FW_l + BW_l\Big) + \frac{D}{B}\sum_{l=1}^{G}(WU_l)$	$2\frac{D}{B}\Big((p-1)(\alpha + \frac{\sum_{l=1}^{G}\lvert w_l\rvert}{p}\delta\beta) +$ $\sum_{l=1}^{G}\Big(2\alpha + B(halo(\lvert x_l\rvert) + halo(\lvert\frac{dL}{dy_l}\rvert))\delta\beta\Big)\Big)$	$\gamma\delta\sum_{l=1}^{G}\Big(2B\frac{(\lvert x_l\rvert+\lvert y_l\rvert)}{p} + 2\lvert w_l\rvert + \lvert bi_l\rvert\Big)$	$p = pw \times ph \leq$ $\min_{l=1}^{G}(W_l \times H_l)$
Layer (Pipeline)	$\frac{D(p+S-1)}{S}\Big(\max_{i=1}^{p}(FW_{G_i})$ $+ \max_{i=1}^{p}(BW_{G_i})\Big) + \max_{i=1}^{p}(WU_{G_i})\Big)$	$2\frac{D(p+S-2)}{B}\Big(\max_{i=1}^{p-1}\big(\alpha + \frac{B}{S}\lvert y_{G_i}\rvert\delta\beta\big)\Big)$	$\gamma\delta\max_{i=1}^{p}\Big(\sum_{l=1}^{G_i}\big(2B(\lvert x_l\rvert + \lvert y_l\rvert)$ $+2\lvert w_l\rvert + \lvert bi_l\rvert\big)\Big)$	$p \leq G$
Filter	$\frac{D}{p}\sum_{l=1}^{G}\Big(FW_l + BW_l\Big) + \frac{D}{Bp}\sum_{l=1}^{G}(WU_l)$	$3\frac{D}{B}(p-1)\sum_{l=1}^{G-1}(\alpha + \frac{B\lvert y_l\rvert}{p}\delta\beta)$	$\gamma\delta\sum_{l=1}^{G}\Big(2B(\lvert x_l\rvert + \lvert y_l\rvert) + \frac{2\lvert w_l\rvert}{p} + \lvert bi_l\rvert\Big)$	$p \leq \min_{l=1}^{G}(F_l)$
Channel	$\frac{D}{p}\sum_{l=1}^{G}\Big(FW_l + BW_l\Big) + \frac{D}{Bp}\sum_{l=1}^{G}(WU_l)$	$3\frac{D}{B}(p-1)\sum_{l=1}^{G-1}(\alpha + \frac{B\lvert y_l\rvert}{p}\delta\beta)$	$\gamma\delta\sum_{l=1}^{G}\Big(2B(\lvert x_l\rvert + \lvert y_l\rvert) + \frac{2\lvert w_l\rvert}{p} + \lvert bi_l\rvert\Big)$	$p \leq \min_{l=1}^{G}(C_l)$
Data + Filter	$\frac{D}{p}\sum_{l=1}^{G}\Big(FW_l + BW_l\Big) + \frac{D}{Bp2}\sum_{l=1}^{G}(WU_l)$	$3\frac{D}{B}(p2-1)\sum_{l=1}^{G-1}(\alpha + \frac{B\lvert y_l\rvert}{p}\delta\beta) +$ $2\frac{D}{B}(p1-1)(\alpha + \frac{\sum_{l=1}^{G}\lvert w_l\rvert}{p}\delta\beta)$	$\gamma\delta\sum_{l=1}^{G}\Big(\frac{2B(\lvert x_l\rvert+\lvert y_l\rvert)}{p1} + \frac{2\lvert w_l\rvert}{p2} + \lvert bi_l\rvert\Big)$	$p = p1 \times p2 \leq$ $B \times \min_{l=1}^{G}(F_l)$

In this work, we focus on the HPC aspects of scaling six different strategies for model and hybrid parallelism in CNNs distributed training. While most works in the literature focus on improving the performance of one single parallelism strategy for one specific framework; our study functions as the basis for a tool, named *ParaDL*, capable of modeling and predicting the performance of a large set of configurations for CNN distributed training at scale. In addition, ParaDL also helps to reveal the practical limits and bottlenecks of different parallel strategies in CNN training .

2 Performance and Memory Projection

In this section we introduce our oracle (*ParaDL*). Through the information that we can get beforehand, such as the dataset, model, supercomputer/cluster system specification, and user's constraints (e.g., maximum number of involved PEs), ParaDL calculates the computation and communication time to project the overall performance. ParaDL can be used for the following purposes:

- Suggesting the best strategy for a given CNN, dataset, and resource budget
- Identifying the time and resources to provision from a system
- Comparison of projections with measured results to detect abnormal behavior
- Identifying limitations of parallel strategies, shortcomings of frameworks, and bottlenecks in systems
- As an education tool of the parallel strategies that would improve the understanding of parallelism in DL

Frameworks that are used for DL are comprised of complex and interleaved layers of optimized functions. A pure analytical model of parallel strategies in CNNs would, therefore, be impractical. In this paper we adopt a hybrid analytical/empirical modeling approach at which we: (i) use analytical modeling for functional requirements driven by the parallelism strategies, and (ii) empirical parametrization for functions not related to the parallel strategy being deployed. We summarize our analytical model in Table 1 . Finally, we quantify the accuracy of the oracle with a large empirical evaluation in Section 3.

3 Evaluation and Results

We conduct a wide range of experiments to show the accuracy and utility of ParaDL in projecting the performance. We choose different CNN models and datasets with different characteristics that affect performance and memory requirements. Experiments are performed on a multi-petaflop supercomputer, with two Intel Xeon Gold 6148 Processors and four NVIDIA Tesla V100 GPUs (16GB of memory per GPU) on each compute node. The GPUs are connected intra-node to the CPUs by PLX switches and PCIe Gen3 x16 links (16 GBps), and together by NVLink (20 GBps). The compute nodes are connected in a 3-level fat-tree topology which has full-bisection bandwidth, and 1:3 over-subscription for intra-rack and inter-rack, respectively (two InfiniBand EDR, e.g., 12.5 GBps, per compute node and 17 compute nodes per rack).

Figure 1 shows the oracle's projections versus the measured runs for different parallel strategies using three different models. The figure is divided in three rows, one for each CNN model, and six columns, one for each parallelism strategy. The parameter b shows the mini-batch size for each case. The x-axis shows the number of GPUs, up to the scaling limit of the specific parallel strategy (e.g., maximum number of filters). More specifically, we scale the tests from 16 to 1024 GPUs for data and hybrid parallelism, from 4 to 64 GPUs for filter/channel parallelism, and up to 4 GPUs for pipeline parallelism. The y-axis shows the iteration time for each case. The iteration time is calculated as an average of 100 iterations excluding the first iteration which normally involves initialization tasks. To get a more detailed analysis, we decompose the execution time into computation and communication. The oracle prediction is shown in blue as stacked bars, i.e., computation+communication, and the measured empirical results are shown in orange. In this figure, we report the best communication times obtained during our experiments, as this represents the peak performance the hardware can deliver and leave aside occasional delays due to external factors such as network congestion coming from other apps, system noise and, overheads due to correctable errors, among others. The labels above each column show the *projection accuracy* in percentage, i.e., 1 - ratio of the absolute value of the difference with respect to the total measured time.

The accuracy of ParaDL predictions for the different parallel strategies are 96.10% for data parallelism, 85.56% for Filter, 73.67% for Channel, 91.43% for Data+Filter, 83.46% for Data+Spatial and 90.22% for pipeline across all CNN models. In general, this represents an overall accuracy of 86.74% for ParaDL, across all parallelism strategies and CNN models, and up to 97.57% for on VGG16.

4 Conclusion

We propose an analytical model for characterizing and identifying the best technique of different parallel strategies for CNN distributed training. We run a wide range of experiments with different models, different parallel strategies and different datasets for up to 1,000s of GPUs and compare with our analytical model. The results demonstrate the accuracy of ParaDL, as high as 97.57% , and 86.74% on average accuracy across all parallel strategies on multiple CNN models and datasets on up to 1K GPUs.

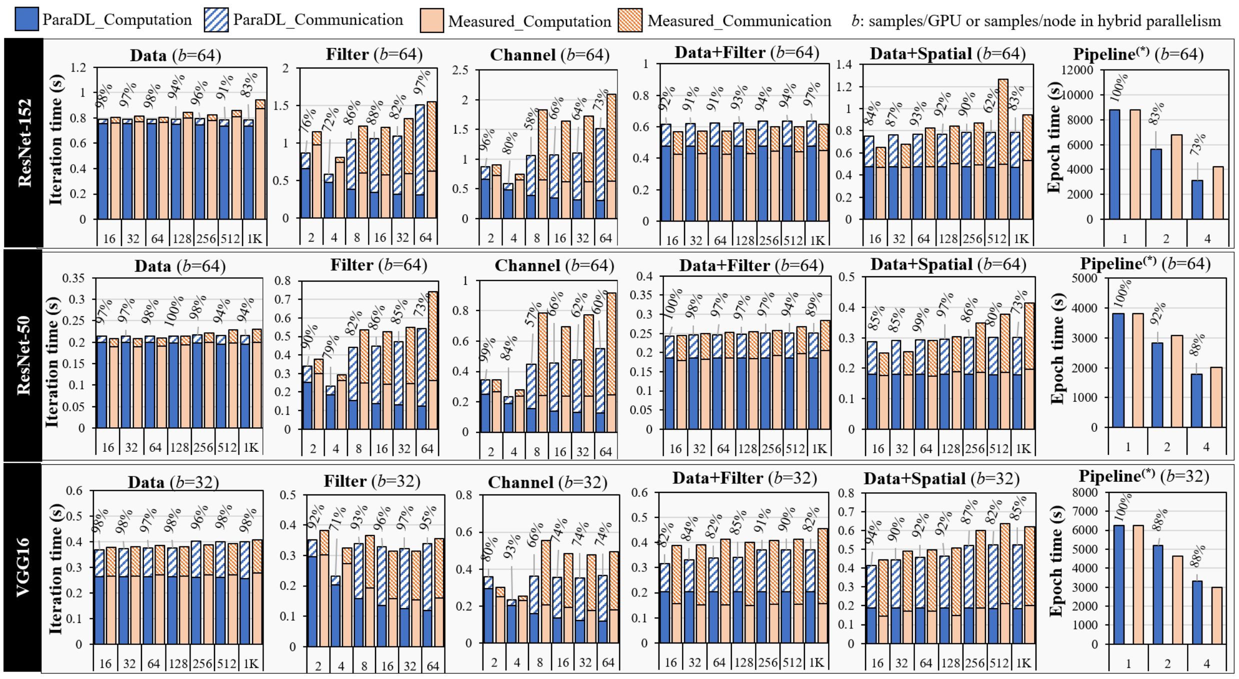

Figure 1: Time breakdown of our analytical model (ParaDL) in comparison with measured runs. The label above each column shows the *projection accuracy*. The x-axis is the number of GPUs. Filter/channel are strong scaling.$^{(*)}$Values are total time since pipeline parallelism [K+20] overlaps the computation and communication.

References

[B+18] Tal Ben-Nun et al. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *CoRR*, abs/1802.09941, 2018.

[H+18] Yanping Huang et al. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. *CoRR*, abs/1811.06965, 2018.

[K+18] Thorsten Kurth et al. Exascale Deep Learning for Climate Analytics. SC '18, pages 51:1–51:12, 2018.

[K+20] Chiheon Kim et al. torchgpipe: On-the-fly pipeline parallelism for training giant models. *arXiv preprint arXiv:2004.09910*, 2020.

[S+14] Yelong Shen et al. Learning semantic representations using convolutional neural networks for web search. WWW '14 Companion, page 373â374, New York, NY, USA, 2014. Association for Computing Machinery.

[W+15] Izhar Wallach et al. Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery, 2015.

[Y+18] Yang You et al. ImageNet Training in Minutes. In *Proceedings of the 47th International Conference on Parallel Processing*, ICPP 2018, pages 1:1–1:10, 2018.