POSTER: Layrub: Layer-centric GPU memory reuse and data migration in extreme-scale deep learning systems

Bo Liu, Wenbin Jiang*, Hai Jin, Xuanhua Shi, and Yang Ma
Services Computing Technology and System Lab, Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
wenbinjiang@hust.edu.cn

Abstract

Growing accuracy and robustness of *Deep Neural Networks* (DNN) models are accompanied by growing model capacity (going deeper or wider). However, high memory requirements of those models make it difficult to execute the training process in one GPU. To address it, we first identify the memory usage characteristics for deep and wide convolutional networks, and demonstrate the opportunities of memory reuse on both intra-layer and inter-layer levels. We then present Layrub, a runtime data placement strategy that orchestrates the execution of training process. It achieves layer-centric reuse to reduce memory consumption for extreme-scale deep learning that cannot be run on one single GPU.

CCS Concepts • **Computer systems organization → Neural networks**; **Heterogeneous (hybrid) systems**;

Keywords Data Placement, DNN, GPU, Memory Efficiency

ACM Reference format:
Bo Liu, Wenbin Jiang*, Hai Jin, Xuanhua Shi, and Yang Ma. 2018. POSTER: Layrub: Layer-centric GPU memory reuse and data migration in extreme-scale deep learning systems. In *Proceedings of ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Vienna, Austria, February 24–28, 2018 (PPoPP'18),* 2 pages.
https://doi.org/10.1145/3178487.3178528

1 Introduction

Deep learning has become one of the most promising techniques in the filed of machine learning. The aforementioned breakthrough is mainly made by two main reasons: large-scale training data and large-scale neural network structures, which interweave with each other. Since DNN architecture is responsible for the quality of model, the design principles of DNNs are worth considering. *Depth* and *width* are acknowledged by almost all researchers [2, 4] as the most important factors for constructing DNNs. The accuracy of models can be significantly increased with steadily increasing depth and width. To be specific, the strength of deep learning has triggered a race to build models going deeper or wider, which leads to extreme-scale DNN models.

However, memory consumption is a main challenge for the extreme-scale deep learning using GPU. Current systems cannot run the extreme-scale models on a single GPU, which cause the unsuccessful training. They can only run the models that fit into the GPU. An approach is required that can handle the extreme-scale models in a more easy manner. There are threee key observations: 1) the memory footprint of the activation data and their gradients are of identical size; 2) the memory footprint between different layers are independent; 3) the activation data is the major contributor to the memory footprint. These properties are suitable for our intra-layer and inter-layer memory reuse strategies, and for maintaining the lowest memory usage for layer-centric computation. The core idea is *reuse*, which is motivated by that the memory space allocated for some data is reused for others if these are generally independent of each other.

2 Design

In this paper, we firstly exploit computation order in the training process, and then introduce two orthogonal strategies to reuse GPU memory space. At last, we hybridize these two strategies, and propose a hybrid layer-centric memory reuse strategy to perfectly accommodate almost all of DNNs.
Exploitation for Computation Order. In order to manage the input or output data placement, we explore the independence among the sequential operations. Taking the simple *Convolutional Neural Networks* (CNN) model in Figure 1 as an example, we observe operations 4 and 5 in the original backward phase. The former is designed to obtain the parameter gradient ∇P of the layer (FC), and the latter one aims to obtain the neuron data gradient ∇FC. These types of operations are interchangeable without breaking the correctness of computation. After the adaptation, the *fine-tuned*

* corresponding author.

PPoPP'18, February 24–28, 2018, Vienna, Austria
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4982-6/18/02…$15.00
https://doi.org/10.1145/3178487.3178528

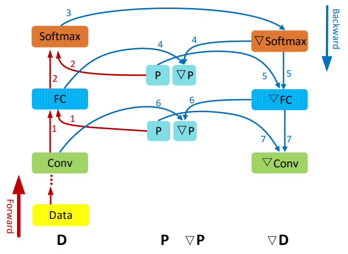

Figure 1. An example of the training process and memory usage characteristics

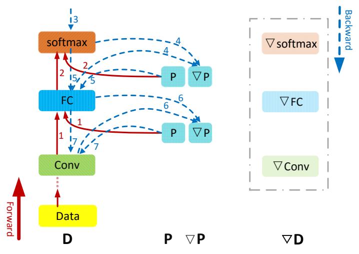

Figure 2. Intra-layer memory reuse back-propagation

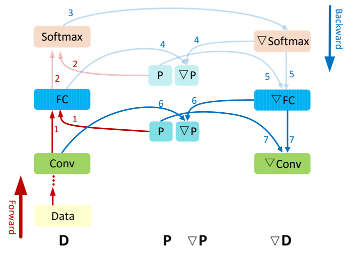

Figure 3. Inter-layer memory reuse and data migration

computation order of the backward phase would be fixed: *the parameter gradient calculation of one layer will take place ahead of neuron gradient calculation of the same layer.*

Intra-Layer Memory Reuse. Based on the computation order, we propose an intra-layer memory reuse strategy. It relies on the operation independence between the forward and backward phases of one layer, and reuses the memory space between them. The data placement could be re-managed: the neuron gradient can reuse the memory space of the neuron data. As illustrated in Figure 2, the operation 1 represents the forward phase of layer (FC), and the operations 4 and 5 both represent the backward phase of layer (FC). The neuron data FC obtained by operation 1 will be firstly utilized by operation 4, and then the memory footprint of FC will be reused by operation 5 to calculate the gradient data ∇FC.

Inter-Layer Memory Reuse and Data Migration. In some cases, where the width of the DNN models is limited but the depth is extremely unbounded, the proportion of memory usage is still unacceptable after using the intra-layer strategy. Therefore, we propose an inter-layer memory reuse strategy. It can reuse a constant amount of memory space for the sequential operations of different layers, and can migrate related data between CPU memory and GPU memory.

3 Preliminary Results

The performance of Layrub is evaluated in the extreme-scale training over GPU. We conduct experiments on a machine with NVIDIA Tesla K40M GPU. We deploy Layrub on the basis of Caffe [3], and modify it using the proposed strategies. The experiments use some well-known models including LeNet, Cifar10_quick, CaffeNet, VGGNet, GoogLeNet, ResNet, WRN; and two famous datasets including Cifar-10 and ImageNet. All configurations follow the default hyper-parameters provided by Caffe and articles.

Memory Efficiency for well-known models. We propose the hybrid memory reuse strategy, which applies the intra-layer strategy on the basis of the inter-layer strategy, making the two strategies complementary to each other. The results in Figure 4 show that the hybrid reuse strategy is suitable for almost types of networks. It can reduce memory usage from 55% to 98.9% during the eight well-known models.

Memory Efficiency for Extreme-scale Models. We take MXNet [1] as a contrast, since it is the only system can train deep models with more than 1001 layer with a single GPU. Figure 5 demonstrates the superiority of Layrub for training extreme-scale models. When the depth of the model (batchsize = 32) increases to 1517, Layrub still makes the training process work well. In contrast, MXNet can train the extra-deep ResNets when the depth of these is no greater than 1172. Our results show that for MXNet, the runtime memory consumption on ResNet-1172 is 11455MB, which is close to the upper boundary of K40M memory of 11580MB.

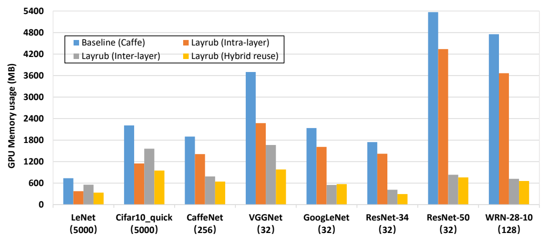

Figure 4. GPU memory allocation for well-known DNN models

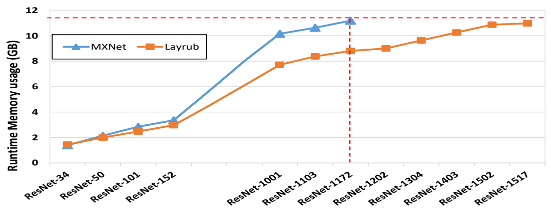

Figure 5. Runtime memory usage on extreme-scale models

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 61672250.

References

[1] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174* (2016).

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. IEEE, Las Vegas, NV, USA, 770–778. https://doi.org/10.1109/CVPR.2016.90

[3] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia (MM'14)*. ACM, Orlando, Florida, USA, 675–678. https://doi.org/10.1145/2647868.2654889

[4] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146* (2016).