Robust Class Parallelism - Error Resilient Parallel Inference with Low Communication Cost

Yaoqing Yang*, Jichan Chung*, Guanhua Wang, Vipul Gupta, Adarsh Karnati, Kenan Jiang
Ion Stoica, Joseph Gonzalez, Kannan Ramchandran
Department of EECS, UC Berkeley, Berkeley, USA
{yqyang, jichan3751, guanhua, vipul_gupta, akarnati, kenanj11, istoica, jegonzal, kannanr}@berkeley.edu

***Abstract*—Model parallelism is a standard paradigm to decouple a deep neural network (DNN) into sub-nets when the model is large. Recent advances in *class parallelism* significantly reduce the communication overhead of model parallelism to a single floating-point number per iteration. However, traditional fault-tolerance schemes, when applied to class parallelism, require storing the entire model on the hard disk. Thus, these schemes are not suitable for soft and frequent system noise such as stragglers (temporarily slow worker machines). In this paper, we propose an erasure-coding based redundant computing technique called *robust class parallelism* to improve the error resilience of model parallelism. We show that by introducing slight overhead in the computation at each machine, we can obtain robustness to soft system noise while maintaining the low communication overhead in class parallelism. More importantly, we show that on standard classification tasks, robust class parallelism maintains the state-of-the-art performance.**

***Index Terms*—Distributed computing, deep learning, system robustness, computation redundancy.**

I. Introduction

The increasing size of cutting-edge models poses a challenge to the limited memory capacity on GPUs. To deal with large models that cannot fit into a single GPU, implementing training and testing using model parallelism [1] is standard practice. A critical drawback of model parallelism, however, is the increased communication cost that arises from both frequent message passing and the large size of messages exchanged (i.e., because the exchanged feature maps can be much larger than network weights). To alleviate this problem, class parallelism [2] pushes the limit of reducing communication overhead by transmitting a single floating point number per machine. The main idea of class-parallelism is to let each partial model only train on a particular class of data. In this way, parallel training and inference can be achieved with minimum communication overhead.

The novel class parallel methods call for new designs for resilience against system noise. For example, One common problem in distributed setups is the existence of persistent stragglers, which are compute nodes that are significantly slower than others due to unpredictable factors impacting their compute and communication time [3]. The straggler problem is even more evident in today's large-scale computing systems; for instance, in serverless computing, about 2% of 3500 commodity machines are observed to be stragglers [4]. Left untreated, these stragglers severely impact latency, as the total execution time depends on the slowest machine.

Another form of system noise is machine *preemption* in elastic compute [5], where inexpensive machines can leave (when preempted) in the middle of a job on a short notice. Thus, longer jobs such as training for large DNNs are vulnerable to this kind of noise. The preemption mechanism is widely applied to cluster management, such as in YARN [6] and Kubernetes [7]. The machines to be preempted are determined by the cluster scheduler and are not known to the tenants in advance [8]. Traditional fault-tolerance schemes such as disk checkpointing require the entire system to wait for the recovery of the failures, causing high disk I/O and long latency.

In this paper, we propose a robust class-parallel scheme for both training and prediction serving. The main idea of our scheme is to introduce redundancy in class parallelism by increasing classes assigned to each machine. In our scheme, we can safely ignore the preemption type of failures (during training) or the slow machines in a system with stragglers, provided that the number of preemptions or stragglers is below a certain threshold determined by the amount of redundancy. Note that redundancy techniques on deep learning have been studied in [9], [10]. Compared to [9], robust class parallelism does not need to wait for a batch of input samples to start prediction serving. It also significantly reduces the communication overhead compared to model parallelism [10].

The primary focus of the paper is on the inference/prediction-serving stage [11]. However, robust class parallelism also applies to the training stage to provide speed-up from parallelism and reduced communication. Using a class-parallel scheme, the distributed training of a partial model at each machine is independent of others, and the preemption or the slow machines only affect the local training. During inference, the model prediction time can be reduced due to parallel gain while providing straggler resiliency, thus resulting in greater speedups. More importantly, robust class parallelism can preserve the classification accuracy even in the precence of system noise, such as stragglers and preemptions.

II. Robust class parallelism

First, we present the class parallelism technique [2] which was designed to reduce communication overhead in model-parallel training of DNNs. Then, we introduce the robust class parallelism and show its robustness to system noise through experiments on popular computer vision datasets.

* Both authors contributed equally.

This preliminary work is an invited paper at Asilomar 2020.

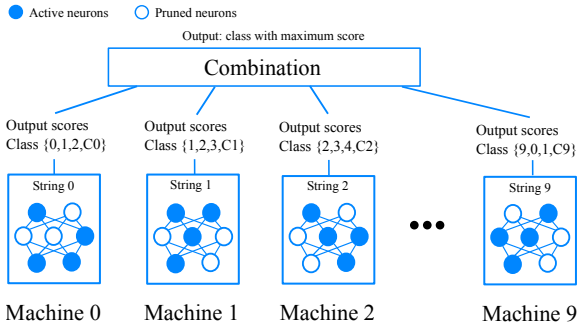

Fig. 1. **(Robust class parallelism)** The figure shows a cyclic class partitioning scheme. The redundant factor is 3, and the final result can be obtained even if 2 machines are unavailable.

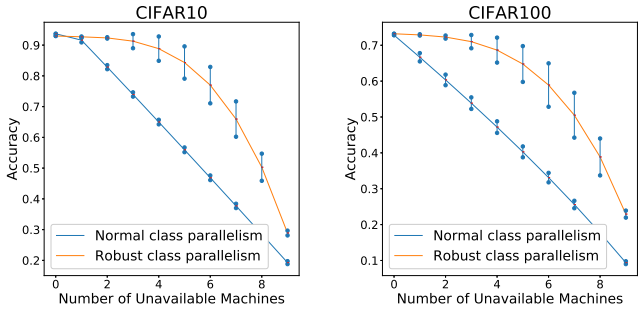

Fig. 2. Test accuracy as a function of the number of machine failures on CIFAR10 and CIFAR100. Both normal and robust class parallelism models are pruned to 50% size.

A. *Class-parallel model partitioning*

Class parallelism can reduce the communication cost while maintaining the small memory footprint in model parallelism. Specifically, the technique has two phases:

- Phase 1: Train the entire model (replicated at each machine) for a short time using data parallelism. Then, assign one class of data to one machine and *prune* the model on that machine based on the activation of the particular class.
- Phase 2: Retrain the partial model on every single machine to be a binary classifier (called a *string* according to [2]), predicting whether the data belongs to the class assigned to this machine or not.

Class parallelism has two advantages: (1) Reduced computation at each machine, coming from class-based data partitioning and network pruning; (2) Reduced communication compared to model parallelism, coming from the fact that decoupled binary classifiers do not communicate among each other. The only single-number communication happens when aggregating the final output class prediction, i.e., each binary classifier outputs a single number regarding the softmax score of whether the input sample belongs to this specific class.

B. *Robust class parallelism*

Here, we introduce *parity strings* which can take more than one classes at each machine. That is, in Phase 1, we prune each string based on the activation of $r > 1$ classes of data, and in Phase 2, we retrain each string using the same redundant data as well. Instead of a single floating-point number, each machine now communicates $r + 1$ numbers during the combination phase, in which the last number denotes the softmax score of the NOT class, the union of all classes not handled at this machine. In Fig. 1, we illustrate a cyclic class partitioning scheme, in which each machine takes $r = 3$ classes. The redundant outputs from the parity strings provide robustness to system noise because when some machine is preempted or is straggling, the results can be aggregated from other replicas.

More specifically, the classification score of the i-th class can be the average from all the available replicas in the presence of system noise. Although assigning a higher redundant factor means higher robustness, it also leads to increased task complexity and more digits to transfer at each machine.

C. *Experiment results*

We compare the robustness of normal and robust class parallelism as a function of unavailable machines. From Fig. 2, robust class parallelism has higher average accuracy than the normal one.

Experiment details: We use a ResNet110 model on CIFAR10 and CIFAR100. Each machine is assigned three classes in robust class parallelism and one class in normal class parallelism. Both the normal and robust class parallelism models are pruned to 50% of the original size. Results on more models will be provided in the full version.

REFERENCES

[1] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *NIPS*, 2012.

[2] G. Wang, Z. Liu, S. Zhuang, B. Hsieh, J. Gonzalez, and I. Stoica, "SensAI: Fast ConvNets Serving on Live Data via Class Parallelism," *MLOps Systems workshop in MLSys*, 2020.

[3] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, pp. 74–80, 2013.

[4] V. Gupta, D. Carrano, Y. Yang, V. Shankar, T. Courtade, and K. Ramchandran, "Serverless straggler mitigation using local error-correcting codes," *arXiv preprint arXiv:2001.07490*, 2020.

[5] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: fair scheduling for distributed computing clusters," in *SOSP*, 2009, pp. 261–276.

[6] *YARN Resource Management*, available at https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.6.5/bk_yarn-resource-management/content/preemption.html.

[7] *Pod Priority and Preemption*, available at https://kubernetes.io/docs/concepts/configuration/pod-priority-preemption/.

[8] B.-G. Chun, T. Condie, Y. Chen, B. Cho, A. Chung, C. Curino, C. Douglas, M. Interlandi, B. Jeon, J. S. Jeong *et al.*, "Apache REEF: Retainable evaluator execution framework," *ACM Transactions on Computer Systems (TOCS)*, vol. 35, no. 2, p. 5, 2017.

[9] J. Kosaian, K. Rashmi, and S. Venkataraman, "Parity models: erasure-coded resilience for prediction serving systems," in *SOSP*, 2019, pp. 30–46.

[10] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover, "A unified coded deep neural network training strategy based on generalized polydot codes," in *ISIT*, 2018, pp. 1585–1589.

[11] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 613–627.