Efficient Resource Sharing for Distributed Deep Learning

Changho Hwang Taehyun Kim Kyuho Son Jinwoo Shin KyoungSoo Park
School of Electrical Engineering, KAIST

Abstract

GPU clusters for training deep learning (DL) models are commonly shared by multiple jobs for efficient utilization of GPUs. Existing cloud resource managers such as Mesos [2], YARN [6], and Borg [7] are widely used for GPU clusters as well, they employ static resource allocation, which often makes it challenging to minimize average job completion time (JCT) and makespan – they do not re-distribute GPUs of a running job for a newly submitted job. In contrast to static allocation that inherently prioritizes first-coming jobs, *dynamic* allocation continuously adjusts the share of all jobs to optimize overall cluster utilization. Dynamic re-adjustment of resource is especially critical for distributed DL jobs whose training throughput often does not increase linearly as the number of distributed resources. Fine-grained adjustment would redistribute the resources to the jobs that can better utilize than others, which ends up improving the JCT and makespan of all jobs. A recent DL job scheduler like Optimus [4] corroborates that dynamic resource allocation greatly helps optimize average JCT and makespan.

However, switching from static to dynamic resource allocation presents a new challenge. It requires each job (user application) to describe how to parallelize their training according to the dynamically-changing set of resources as they lose or gain GPUs. Unfortunately, this makes user application much more complex and difficult to optimize because it should carefully consider the capacity of hardware resources and their configuration. For instance, the network topology of allocated GPUs (e.g. all connected via PCIe, P2P connected via NVLink, part of them are in another machine connected via Ethernet, etc) has a large impact on where to update each parameter and how to aggregate gradients from GPUs and re-distribute the updated parameters. An optimal decision often requires online measurements to figure out a good load balancing scheme across parallel devices, which should be repeated whenever the resource manager makes a new allocation decision for the job. It will be even more complicated if we take heterogeneous resources into consideration as a new lineup of GPU comes out almost once every year. Existing auto-parallelization frameworks such as Horovod [5], Parallax [3], and PaddlePaddle [1] cannot salvage the situation as they assume a fixed and homogeneous set of devices on a specific network topology.

In this work, we propose an efficient design of DL training system that *separates* dynamic resource management from user applications. Instead, each user declares training of a DL model as a data-flow graph and submits it to a system framework, then the system dynamically decides resource share of each model and performs parallelized training automatically. Such a design not only frees users from complex parallelization and optimization but also lets the system handle these challenges much efficiently using two key designs as follows. First, we present an accurate performance prediction model of a DL job with an arbitrarily given networking topology available in the cluster. This prediction model lets the resource manager make a more accurate decision with awareness of the inefficiency from the network topology, that helps it optimize overall cluster throughput. When a new job is submitted, our prediction model performs offline inspection of every data transaction required for synchronization and estimates their expected finishing time to calculate overall synchronization overhead of the model with the given network topology. For more accurate and faster estimation, the system collects data transaction performance from online execution history across multiple jobs over various inter-GPU links to be used in future estimations. This system-side estimation is more accurate and efficient than user-side approach because it utilizes every information collected from earlier job executions. Second, we design a new auto-parallelization framework specialized for dynamic and heterogeneous resource allocation. While existing frameworks suffer from large restarting overhead to change the set of devices to use, our framework performs it efficiently during runtime without restarting the training. It also efficiently utilizes parallel heterogeneous devices via 1) careful decision of the device to update each parameter considering data transaction and parameter updating throughput and 2) online load balancing of the batch size to be processed on different GPUs by setting back-propagation over those GPUs to be finished at the same time that maximizes GPU utilization. Our evaluations show that our auto-parallelization shows similar or even better throughput comparing with manually parallelized training code written by experts while reducing lines of code of user applications as well.

References

[1] PaddlePaddle. http://paddlepaddle.org/.

[2] HINDMAN, B., KONWINSKI, A., ZAHARIA, M., GHODSI, A., JOSEPH, A. D., KATZ, R. H., SHENKER, S., AND STOICA, I. Mesos: A platform for fine-grained resource sharing in the data center. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (2011).

[3] KIM, S., YU, G.-I., PARK, H., CHO, S., JEONG, E., HA, H., LEE, S., JEONG, J. S., AND CHUN, B.-G. Parallax: Automatic data-parallel training of deep neural networks. *CoRR abs/1808.02621* (2018).

[4] PENG, Y., BAO, Y., CHEN, Y., WU, C., AND GUO, C. Optimus: An efficient dynamic resource scheduler for deep learning clusters. In *Proceedings of the ACM European Conference on Computer Systems (EuroSys)* (2018).

[5] SERGEEV, A., AND BALSO, M. D. Horovod: fast and easy distributed deep learning in tensorflow. *CoRR abs/1802.05799* (2018).

[6] VAVILAPALLI, V. K., MURTHY, A. C., DOUGLAS, C., AGARWAL, S., KONAR, M., EVANS, R., GRAVES, T., LOWE, J., SHAH, H., SETH, S., SAHA, B., CURINO, C., O'MALLEY, O., RADIA, S., REED, B., AND BALDESCHWIELER, E. Apache Hadoop YARN: yet another resource negotiator. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC)* (2013).

[7] VERMA, A., PEDROSA, L., KORUPOLU, M., OPPENHEIMER, D., TUNE, E., AND WILKES, J. Large-scale cluster management at google with borg. In *Proceedings of the ACM European Conference on Computer Systems (EuroSys)* (2015).