SLO-Aware Space-Time GPU Sharing for DL Workloads

Ubaid Ullah Hafeez, Anshul Gandhi
{uhafeez,anshul}@cs.stonybrook.edu
Stony Brook University, Stony Brook, NY, USA

Abstract

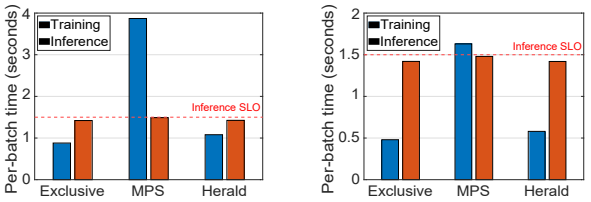
GPU-equipped servers are extensively used for Deep Learning (DL) jobs to train and serve Deep Neural Nets (DNNs) for inference [2, 4]. However, for a given DL job, it is often infeasible to fully utilize allocated GPUs even when employing state-of-the-art DL frameworks [6–8, 11, 14]. It has been observed that DNN execution can result in under-utilization of both memory and compute resources offered by present day GPUs [5, 20, 21]. This is primarily due to the disparity in the capability of available GPUs and the requirements of DL jobs—commodity GPUs are available in a few fixed configurations [6] whereas modern DL jobs exhibit immense variety in their compute and memory requirements. This mismatch between the resource requirements of DL jobs and the capabilities of available GPUs presents an opportunity to increase eficiency by sharing GPUs among multiple jobs.

Prior works [3, 10, 13, 15, 16, 19, 20] explore various techniques to maximize GPU utilization via sharing among multiple DL jobs. While effective, most of the prior works ignore SLOs of the underlying jobs. In addition, prior works [5, 21] typically explore scenarios in which GPU resources are either exclusively shared by multiple training jobs or exclusively shared by multiple inference jobs, overlooking the mixed-use case where training jobs may be colocated with inference jobs on the same set of GPU resources. This mixed-use case is particularly appealing for resource eficiency given the contrasting characteristics of training and inference jobs. Inference jobs typically are end-user-facing and are therefore latency sensitive, unlike training jobs.

Our work, Herald addresses the main challenges involved in sharing GPUs among training and inference jobs. Herald does this by implementing algorithms to enable careful spatial- and time-sharing of GPUs between training and inference jobs. Our algorithms adhere to the SLO requirements of DL jobs while increasing the eficiency of GPU resources.

To design spatial- and time-sharing techniques, Herald makes use of two key observations. First, a DL job typically does not use GPU compute resources continuously. There are periods during which the GPUs are either idle or partially utilized. Second, there are times when a DL job fully utilizes GPUs' compute resources, leaving no room for sharing of resources without sacrificing job performance.

Herald estimates requirements for all the compute operations using the underlying data flow graph of the DL jobs. This enables Herald to identify "light" operations for which GPU can be shared without any significant performance degradation. For the more compute-intensive "heavy" operations where spatial sharing is ineffective and can degrade performance, Herald implements fine-grained time sharing of GPUs to prioritize SLO-sensitive (inference) workloads.



(a) Transformer [18] inference with RNNLM [22] training job (b) Transformer [18] inference with VGG16 [17] training job

Figure 1. Experimental evaluation results for Herald.

We implement Herald by integrating directly into the TensorFlow source code to avoid modifications in user-level code. For scheduling and spatial sharing decisions, Herald employs information about the data-flow graph of each DL job, which can be obtained ofline without much overhead [7, 9, 11]. We also design a new logging tool which only logs infor-mation pertinent to our algorithms, reducing the logging overhead by an order of magnitude compared to the existing `tf.Profiler` [1] in TensorFlow. We use shared memory to keep track of the state of the GPUs and job priorities.

We experimentally evaluate Herald on a server equipped with an NVIDIA Tesla V100 16GB GPU. We share the GPU between two DL jobs: (1) a high priority inference job for a Transformer [18] model, and (2) a low priority training job that is RNNLM [12, 23] in Figure 1(a) and VGG-16 [17] in Figure 1(b), respectively. The inference job has a request rate of 10 samples/second with an SLO of 1.6 seconds to process one batch of data (red dotted line in Figure 1).

In Figure 1, "Exclusive" shows the performance achieved by the jobs when run in isolation (without sharing of GPU resources). We use MPS [3], the default GPU sharing tool by NVIDIA, as the baseline for our performance evaluation. We tried various configurations of MPS to prioritize execution of inference jobs and report numbers for the setting which maintains SLOs for the inference job. However, the colocated training job experiences on average 2.5× increase in per-batch time as compared to "Exclusive". On the other hand, the scheduling and sharing algorithms of Herald ensure that the SLO for the inference job is achieved and that the training job can maximize utilization of GPU resources during times when inference is unable to saturate GPU resources. As a result, per-batch time of training job increases by only 22%, on average, which is 10× lower than that under MPS.

Acknowledgment

This work was supported in part by NSF grants 1730128, 1750109, 2106434, and 2214980.

References

[1] 2020. TensorFlow Profiler. Retrieved Oct 2, 2022 from https://github.com/tensorflow/profiler

[2] 2022. AI Composability and Virtualization: Mellanox Network Attached GPUs. Retrieved Oct 2, 2022 from https://network.nvidia.com/related-docs/solutions/SB_ai_composability_virtualization.pdf

[3] 2022. NVIDIA. CUDA Multi-Process Service. Retrieved Oct 2, 2022 from https://docs.nvidia.com/deploy/mps/index.html

[4] Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodík, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, and Sudipta Sinha. 2017. Real-time video analytics: The killer app for edge computing. computer 50, 10 (2017), 58–67.

[5] Aditya Dhakal, Sameer G Kulkarni, and KK Ramakrishnan. 2020. Gslice: controlled spatial sharing of gpus for a scalable inference platform. In Proceedings of the 11th ACM Symposium on Cloud Computing. 492–506.

[6] Ubaid Ullah Hafeez and Anshul Gandhi. 2020. Empirical Analysis and Modeling of Compute Times of CNN Operations on AWS Cloud. In 2020 IEEE International Symposium on Workload Characterization (IISWC). IEEE, 181–192.

[7] Ubaid Ullah Hafeez, Xiao Sun, Anshul Gandhi, and Zhenhua Liu. 2021. Towards optimal placement and scheduling of DNN operations with Pesto. In Proceedings of the 22nd International Middleware Conference. 39–51.

[8] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. Advances in neural information processing systems 32 (2019).

[9] Beomyeol Jeon, Linda Cai, Pallavi Srivastava, Jintao Jiang, Xiaolan Ke, Yitao Meng, Cong Xie, and Indranil Gupta. 2020. Baechi: fast device placement of machine learning graphs. In Proceedings of the 11th ACM Symposium on Cloud Computing. 416–430.

[10] Myeongjae Jeon, Shivaram Venkataraman, Amar Phanishayee, Junjie Qian, Wencong Xiao, and Fan Yang. 2019. Analysis of {Large-Scale}{Multi-Tenant}{GPU} Clusters for {DNN} Training Workloads. In 2019 USENIX Annual Technical Conference (USENIX ATC 19). 947–960.

[11] Zhihao Jia, Matei Zaharia, and Alex Aiken. 2019. Beyond data and model parallelism for deep neural networks. SysML 2019 (2019).

[12] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410 (2016).

[13] Gangmuk Lim, Jeongseob Ahn, Wencong Xiao, Youngjin Kwon, and Myeongjae Jeon. 2021. Zico: Efficient {GPU} Memory Sharing for Concurrent {DNN} Training. In 2021 USENIX Annual Technical Conference (USENIX ATC 21). 161–175.

[14] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. 2019. PipeDream: generalized pipeline parallelism for DNN training. In Proceedings of the 27th ACM Symposium on Operating Systems Principles. 1–15.

[15] Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. 2021. {INFaaS}: Automated Model-less Inference Serving. In 2021 USENIX Annual Technical Conference (USENIX ATC 21). 397–411.

[16] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. 2019. Nexus: A GPU cluster engine for accelerating DNN-based video analysis. In Proceedings of the 27th ACM Symposium on Operating Systems Principles. 322–337.

[17] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems. 5998–6008.

[19] Xiaofeng Wu, Jia Rao, Wei Chen, Hang Huang, Chris Ding, and Heng Huang. 2021. SwitchFlow: preemptive multitasking for deep learning. In Proceedings of the 22nd International Middleware Conference. 146–158.

[20] Wencong Xiao, Shiru Ren, Yong Li, Yang Zhang, Pengyang Hou, Zhi Li, Yihui Feng, Wei Lin, and Yangqing Jia. 2020. {AntMan}: Dynamic Scaling on {GPU} Clusters for Deep Learning. In 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). 533–548.

[21] Peifeng Yu and Mosharaf Chowdhury. 2019. Salus: Fine-grained gpu sharing primitives for deep learning applications. arXiv preprint arXiv:1902.04610 (2019).

[22] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014).

[23] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014).