Majula: On the Suitability of Spatial Multiplexing Strategies for GPU Accelerators in DNN Model Serving Systems

Anirudh Canumalla

A master's thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

University of Washington

2022

Supervisor: Arvind Krishnamurthy

Program Authorized to Offer Degree:

Paul G. Allen School of Computer Science and Engineering

University of Washington

Abstract

Majula: On the Suitability of Spatial
Multiplexing Strategies for GPU Accelerators
in DNN Model Serving Systems

Anirudh Canumalla

This sample dissertation is an aid to students who are attempting to format their theses with LaTeX, a sophisticated text formatter widely used by mathematicians and scientists everywhere.

- It describes the use of a specialized macro package developed specifically for thesis production at the University. The macros customize LaTeX for the correct thesis style, allowing the student to concentrate on the substance of his or her text.[1]

- It demonstrates the solutions to a variety of formatting challenges found in thesis production.

- It serves as a template for a real dissertation.

[1]See Appendix A to obtain the source to this thesis and the class file.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

ACKNOWLEDGMENTS

I would like to express my deepest thanks to all those who have helped throughout my educational journey. Thank you to my research mentor Arvind Krishnamurthy for his mentorship and guidance. I started systems research late in my undergrad, and Arvind was the best mentor I could have asked for. He always makes one-on-one time for his students and has excellent research vision. He inspires me to be a better researcher, teacher, and computer scientist.

I would like to thank the members of the UW Systems Lab for their insight and feedback throughout this process. In particular, thank you to Lequn Chen for always helping me with whatever questions I had and for providing significant guidance on this project and others. I also appreciate Xiangfeng Zhu for sharing advice and resources. Thank you so much to Tom Anderson and Simon Peter for your mentorship, guidance, and wisdom.

Thank you so much to my friends and family. I wouldn't be here without all of your support. In particular, thank you to my parents Anu and Sridhar for giving me unconditional love and support even when I mess up. Thank you to my brothers and sister in arms; Ashika, Avinash, Santhu, Sumanth and Vishal.

Chapter 1

INTRODUCTION

In the past two decades, advancements in datacenter systems and the proliferation of personal computing devices like smartphones have motivated ever increasing numbers of applications and services to move to the cloud. To date, cloud operators have built their datacenters with commodity components in order to provide large-scale services and gain the benefits of economies of scale. However, with the slowdown of Moore's law and Dennard scaling [4], cloud systems increasingly offload application-specific workloads from the CPU to specialized hardware in order to improve performance, energy usage, and cost-efficiency. In particular, machine learning (ML) is a key example of an emerging and popular workload that benefits from being run on clusters of accelerators, such as graphics processing units (GPU). Broadly, there are two major categories of machine learning systems: systems for ML training, which is the process of learning patterns from data, and model serving systems for ML inference, which is the process of efficiently executing pre-trained models. In the setting of a cloud-scale inference service, we assume clients upload latency-sensitive requests.

Cloud operators codify the latency-sensitivity of inference jobs into serving systems by allowing clients to specify service-level objectives (SLOs), a sort of contract that the service will complete a request in a certain time. Ultimately, the goals of inference systems are to maximize throughput while taking great care to satisfy all SLOs. For this study, we focus on model serving systems for running Deep Neural Network (DNN) models efficiently on GPUs. GPUs process DNNs orders of magnitude faster than CPUs, but are also more expensive. In order for GPUs to be cost-effective, it is essential that cloud operators build model serving systems that are resource-efficient by keeping GPUs at sustained high utilization. In addition to *resource-efficiency*, Jain et al. identify *latency predictability* and *performance isolation* as

key design criterion for model serving systems [14]. These two additional metrics are essential since the goal of model serving systems is to guarantee quality of service by meeting very tight SLOs (typically specified on an order of magnitude between 10^1 (ms) and 10^2 (ms)).

Traditional inference serving systems like Clipper [7] or TFServing [15] utilize an exclusive-access model, where each DNN model has full ownership of a GPU and its resource. This model, although good for performance isolation, leads to very low GPU utilization and, accordingly, poor resource-efficiency. Furthermore, these first generation systems fail on metrics of latency predictability. State-of-the-art systems like Clockwork, Nexus, and Symphony [8, 17, 5] employ time multiplexing strategies where a cluster scheduler interleaves requests for different models such that each GPU only executes requests for a single model in any given insant of time.

Another approach to GPU multiplexing is spatial multiplexing, where the cluster scheduler employs GPU multi-tenancy and concurrently executes multiple DNN workloads. The chief promise of spatial multiplexing is greater resource-efficiency. However, existing approaches for spatial multiplexing severely compromise predictability and performance isolation as demonstrated in Chapter 1.

go to

Chapter 2

BACKGROUND

2.1 Stuff

2.2 more stuff

2.2.1 more stuff gone wrong

Chapter 3

THE THESIS UNFORMATTED

3.1 The Control File

Figure 3.1 shows a control file that might have produced this thesis. It sets the document style, with options and parameters, and formats the various parts of the thesis—but contains no text of its own.

```
% LaTeX thesis control file

\documentclass [11pt, proquest]{uwthesis}[2014/11/13]

\begin{document}
...
\end{document}
```

Figure 3.1: A thesis control file (`thesis.tex`). This file is the input to LaTeX that will produce a thesis. It contains no text, only commands which direct the formatting of the thesis.

Chapter 4

RUNNING LaTeX
(*AND PRINTING IF YOU MUST*)

BIBLIOGRAPHY

[1] Nvidia ampere architecture whitepaper. https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf, 2020.

[2] NVIDIA A100. https://www.nvidia.com/en-us/data-center/a100/, 2022.

[3] NVIDIA Multi-Instance GPU User Guide. https://docs.nvidia.com/datacenter/tesla/pdf/NVIDIA_MIG_User_Guide.pdf, 2022.

[4] Luiz Andr Barroso, Jimmy Clidaras, and Urs Hlzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool Publishers, 2nd edition, 2013.

[5] Lequn Chen, Weixin Deng, Anirudh Canumalla, Yu Xin, Matthai Philipose, and Arvind Krishnamurthy. Symphony: Combining load balancing and scheduling for serving dnns. 2022.

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[7] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In Aditya Akella and Jon Howell, editors, *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 613–627. USENIX Association, 2017.

[8] Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. Serving dnns like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 443–462. USENIX Association, November 2020.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[11] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

[12] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[13] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[14] Paras Jain, Xiangxi Mo, Ajay Jain, Harikaran Subbaraj, Rehan Sohail Durrani, Alexey Tumanov, Joseph Gonzalez, and Ion Stoica. Dynamic space-time scheduling for GPU inference. *CoRR*, abs/1901.00041, 2019.

[15] Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke. Tensorflow-serving: Flexible, high-performance ML serving. *CoRR*, abs/1712.06139, 2017.

[16] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[17] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. Nexus: a GPU cluster engine for accelerating dnn-based video analysis. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP 2019, Huntsville, ON, Canada, October 27-30, 2019*, pages 322–337. ACM, 2019.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[19] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[20] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[21] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[22] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. *CoRR*, abs/2104.00298, 2021.

[23] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.

Appendix A

WHERE TO FIND THE FILES

The uwthesis class file, `uwthesis.cls`, contains the parameter settings, macro definitions, and other TEXnical commands which allow LATEX to format a thesis. The source to the document you are reading, `uwthesis.tex`, contains many formatting examples which you may find useful. The bibliography database, `uwthesis.bib`, contains instructions to BibTeX to create and format the bibliography. You can find the latest of these files on:

- My page.

  `https://staff.washington.edu/fox/tex/thesis.shtml`

- CTAN

  `http://tug.ctan.org/tex-archive/macros/latex/contrib/uwthesis/`

  (not always as up-to-date as my site)