The Anatomy of an Efficient Blackwell GEMM

Antonio Moral Villarín

Table of contents

Abstract **3**

Acknowledgements **3**

List of Figures and Tables **3**

1 Chapter 1 – Introduction **5**
1.1 Motivation and Context: The Need for Hardware–Software Co-Design 5
1.2 Challenges in Efficient Compute for AI and Edge Applications 5
1.3 Objectives and Scope of the Thesis 5
1.4 Methodology Overview 5
1.5 Structure of the Thesis 5

2 Chapter 2 – Background and Related Work **5**
2.1 Evolution of GPU Architectures: From Volta to Blackwell 5
2.2 Hardware–Software Co-Design: Principles and Applications 5
2.3 General Matrix-Matrix Multiplication (GEMM) in AI Workloads 5
2.4 Domain-Specific Languages (DSLs) for GPU Programming 5
2.5 Relevant Publications and Tools (NVIDIA Research, Citadel, JAX Scaling Book, etc.) 5

3 Chapter 3 – Architecture Comparison: Hopper vs Blackwell **5**
3.1 Overview of Hopper Architecture 5
3.2 Overview of Blackwell Architecture 5
3.3 Key Innovations in Blackwell 5
3.3.1 Ultra Tensor Cores and New Precision Formats (FP8, FP4) 5
3.3.2 Transformer Engine and FP4 Micro Scaling 5
3.3.3 Multi-Die Chip Design and Interconnect (NVLink, NVSwitch) 5
3.3.4 Memory System: HBM3e, L2 Cache, and Shared Memory 5
3.4 Performance/Watt and Area Efficiency Considerations 5

3.5 Summary of Architectural Differences 5

4 Chapter 4 – Metrics for GPU Efficiency 5

4.1 Performance per Watt 5

4.2 Compute Throughput by Data Type 5

4.3 Memory Bandwidth and Arithmetic Intensity 5

4.4 Power, Thermal Design, and Silicon Area Constraints 5

4.5 Efficiency Bottlenecks: From Memory Bound to Compute Bound 5

5 Chapter 5 – Programming Models for Modern GPUs 5

5.1 Introduction to GPU DSLs for Performance 5

5.2 Triton 5

5.3 ThunderKittens (TK) 5

5.4 TileLang 5

5.5 Cute and CUTLASS 5

5.6 Gluon 5

5.7 Pallas and the JAX ML Scaling Framework 5

5.8 Summary: DSLs as Enablers of Architectural Efficiency 5

6 Chapter 6 – Methodology and Experimental Setup 5

6.1 Objectives of Benchmarking 5

6.2 Hardware Platforms and Specifications 5

6.2.1 Hopper H100 5

6.2.2 Blackwell B200 5

6.3 Software Tools and Libraries Used 5

6.4 Microbenchmark Design: GEMM Kernel Implementations 5

6.5 Measurement Techniques 5

6.5.1 Throughput (FLOP/s) 5

6.5.2 Power Consumption and Efficiency 5

6.5.3 Memory Bandwidth 5

6.6 Ensuring Fairness and Reproducibility 5

7 Chapter 7 – Results and Discussion 5

7.1 Performance Comparison Across Data Types 5

7.2 Analysis of Performance per Watt 5

7.3 Memory Bandwidth Observations 5

7.4 Impact of TMA (Tensor Memory Accelerator) 5

7.5 Roofline Analysis: Compute vs Memory Bound 5

7.6 Real-World Relevance: Case Study on Transformer Inference/Training 5

7.7 Discussion of Bottlenecks and Architectural Impact 5

8 Chapter 8 – Conclusions and Future Work 5

8.1 Summary of Findings 5

8.2 Implications for Hardware–Software Co-Design 5
8.3 Relevance to Edge Computing 5
8.4 Future Work and Doctoral Research Directions 5

9 References 5

10 Appendices 5
10.1 Appendix A: Experimental Scripts and Kernel Listings 5
10.2 Appendix B: Extended Benchmark Results 5
10.3 Appendix C: TMA and GEMM Intrinsics Documentation 5

Abstract

Acknowledgements

List of Figures and Tables

To be auto-generated by Quarto.

1 Chapter 1 – Introduction

1.1 Motivation and Context: The Need for Hardware–Software Co-Design

1.2 Challenges in Efficient Compute for AI and Edge Applications

1.3 Objectives and Scope of the Thesis

1.4 Methodology Overview

1.5 Structure of the Thesis

2 Chapter 2 – Background and Related Work

2.1 Evolution of GPU Architectures: From Volta to Blackwell

2.2 Hardware–Software Co-Design: Principles and Applications

2.3 General Matrix-Matrix Multiplication (GEMM) in AI Workloads

2.4 Domain-Specific Languages (DSLs) for GPU Programming

2.5 Relevant Publications and Tools (NVIDIA Research, Citadel, JAX Scaling Book, etc.)

3 Chapter 3 – Architecture Comparison: Hopper vs Blackwell

3.1 Overview of Hopper Architecture

3.2 Overview of Blackwell Architecture

3.3 Key Innovations in Blackwell

3.3.1 Ultra Tensor Cores and New Precision Formats (FP8, FP4)

3.3.2 Transformer Engine and FP4 Micro Scaling

3.3.3 Multi-Die Chip Design and Interconnect (NVLink, NVSwitch)

3.3.4 Memory System: HBM3e, L2 Cache, and Shared Memory

3.4 Performance/Watt and Area Efficiency Considerations

3.5 Summary of Architectural Differences

4 Chapter 4 – Metrics for GPU Efficiency

4.1 Performance per Watt

4.2 Compute Throughput by Data Type