

Programming Massively Parallel Processors, 1st Edition : A Hands-on Approach
Kirk, David
ISBN: 9780123814722

Chapter 1: Introduction

- 1.1 GPUs as Parallel Computers
- 1.2 Architecture of a Modern GPU
- 1.3 Why More Speed or Parallelism?
- 1.4 Parallel Programming Languages and Models
- 1.5 Overarching Goals
- 1.6 Organization of the Book

Chapter 2: History of GPU Computing

- 2.1. Evolution of Graphics Pipelines
 - The Era of Fixed Function Graphics Pipeline
 - Evolution of Programmable Real-Time Graphics
 - Unified Graphics and Computing Processors
- 2.2. GPGPU: an Intermediate Step
 - Scalable GPUs
 - Recent Developments
 - Future Trends

Chapter 3: Introduction to CUDA

- 3.1. Data Parallelism
- 3.2. CUDA Program Structure
- 3.3. A Matrix-Matrix Multiplication Example
- 3.4. Device Memories and Data Transfer

3.5. Kernel Functions and Threading

3.6. Summary

Function Declarations

Kernel Launch

Predefined Variables

Runtime API

Chapter 4: CUDA Threads

4.1. CUDA Thread Organization

4.2. More on BlockIdx and ThreadIdx

4.3. Synchronization and Transparent Scalability

4.4. Thread Assignment

4.5. Thread Scheduling and Latency Tolerance

4.6. Summary

Chapter 5: CUDA Memories

5.1. Importance of Memory Access Efficiency

5.2. CUDA Device Memory Types

5.3. A Strategy for Reducing Global Memory Traffic

5.4. Memory as a Limiting Factor to Parallelism

5.5. Summary

Chapter 6: Performance Considerations

6.1. More on Thread Execution

6.2. Global Memory Bandwidth

6.3. Dynamic Partitioning of SM Resources

- 6.4. Data Prefetching
- 6.5. Instruction Mix
- 6.6. Thread Granularity
- 6.7. Measured Performance and Summary

Chapter 7: Floating-Point Considerations

7.1. Floating-Point Format

Normalized representation of M

Excess encoding of E

7.2. Representable Numbers

7.3. Special Bit Patterns and Precision

7.4. Arithmetic Accuracy and Rounding

7.5. Algorithm Considerations

7.6. Summary

Chapter 8: Application Case Study I - Advanced MRI Reconstruction

8.1. Application Background

8.2. Iterative Reconstruction

8.3. Computing $F^H d$

Step 1: Determine the Kernel Parallelism Structure

Step 2: Getting Around the Memory Bandwidth Limitation

Step 3: Use Hardware Trigonometry Functions

Step 4: Experimental Performance Testing

8.4. Final Evaluation

Chapter 9: Application Case Study II - Molecular Visualization and Analysis

- 9.1. Application Background
- 9.2. A Simple Kernel Implementation
- 9.3. Instruction Execution Efficiency
- 9.4. Memory Coalescing
- 9.5. Additional Performance Comparisons
- 9.6. Using Multiple GPUs

Chapter 10: Parallel Programming and Computational Thinking

- 10.1. Goals of Parallel Programming
- 10.2. Problem Decomposition
- 10.3. Algorithm Selection
- 10.4. Computational Thinking

Chapter 11: A Brief Introduction to OpenCL™

- 11.1. Background
- 11.2. Data Parallelism Model
- 11.3. Device Architecture
- 11.4. Kernel Functions
- 11.5. Device Management and Kernel Launch
- 11.6. Electrostatic Potential Map in OpenCL
- 11.7. Summary

Chapter 12: Conclusion and Future Outlook

- 12.1. Goals Revisited
- 12.2. Memory Architecture Evolution
- 12.3. Kernel Execution Control Evolution

12.4. Core Performance

12.5. Programming Environment

12.6. A Bright Outlook

Appendix A: Matrix Multiplication Example Code

Appendix B: Speed and feed of current generation CUDA devices