

课程大作业二基础题：GPU学习智能体（agent）

- ✓ 选择一个赛道：摩尔、沐曦、并行；
- ✓ 准备数据集：
 - ✓ **GPU编程问答对**：基于本年度GPU编程教材（ Programming Massively Parallel Processors: A Hands-on Approach, 2017）生成问答对；
 - ✓ 基础题：教材的蓝色画勾部分
 - ✓ 加分题：教材的红色画勾部分

课程大作业二基础题：GPU学习智能体（agent）

- ✓ 选择一种训练好的开源大模型：Llama、Qwen、DeepSeek、等；
- ✓ 选择一种大模型微调框架：
 - ✓ 摩尔赛道：Llama Factory、等；
 - ✓ 沐曦赛道：Unsloth、等；
 - ✓ 并行赛道：Unsloth、等；

课程大作业二基础题: GPU学习智能体 (agent)

- ✓ 采用低秩适应 (Low-Rank Adaptation, LoRA)，通过大模型微调框架，基于你编写的 GPU编程问答对，对你选的开源大模型进行微调，获得微调后的**大模型文件 (Hugging Face Transformers 标准格式)**；

课程大作业二：GPU学习智能体（agent）

- ✓ 选择一种推理框架：vllm、SGLang、等，让你的**GPU学习智能体**工作起来哦；
 - ✓ 基础题：非必须用推理框架；
 - ✓ 加分题：采用一种推理框架。

课程大作业二加分题: GPU学习智能体 (agent)

✓ 所有有利于提高考核指标成绩的方法，包括但不限于：

- ✓ 采用RAG、SFT、PPO等方法，提高大模型微调的质量，提高准确率；
- ✓ 采用你在本课程上学到的GPU编程的语言，包括：CUDA、PTX、Triton、TileLang、等，加速推理框架的算子，提高速度；
- ✓ 可以做一个UI界面，让你的GPU学习智能体更吸引人；
- ✓ 可以给你的GPU学习智能体取一个响亮的名字。

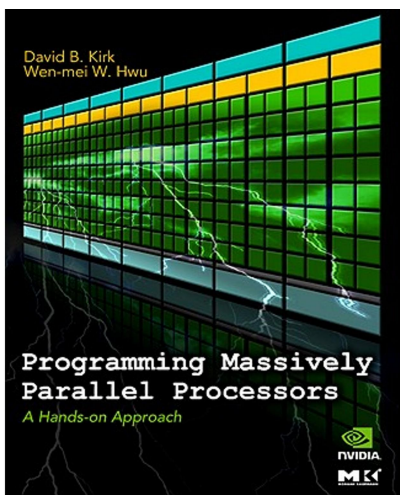
课程大作业二考核方式: GPU学习智能体 (agent)

- ✓ 大作业二提交方式: 将代码和一份描述文档, 上传到本课程gitee, 以便评分。

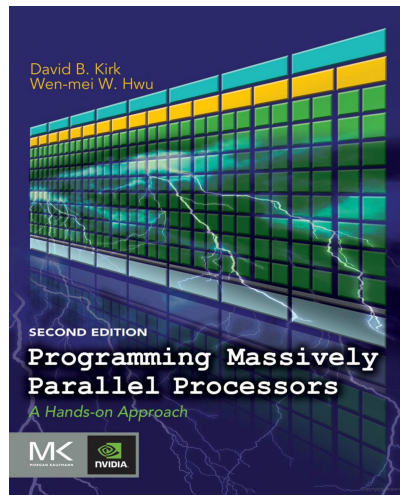
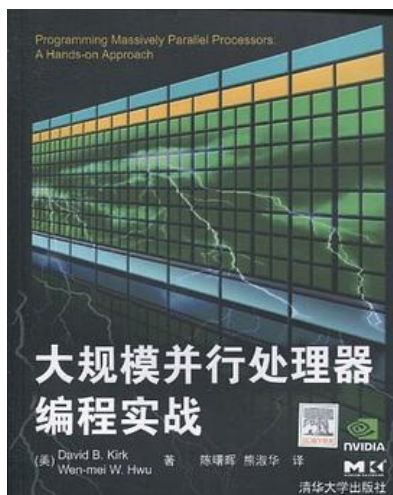
课程大作业二考核方式:GPU学习智能体 (agent)

- ✓考核指标：速度 (tokens/s)、准确率；
- ✓速度 (tokens/s)：每个赛道的服务器测试的结果；
- ✓准确率：
 - ✓基础题：给定的测试集（仅限教材蓝色画勾的部分）；
 - ✓加分题：评委的打分。

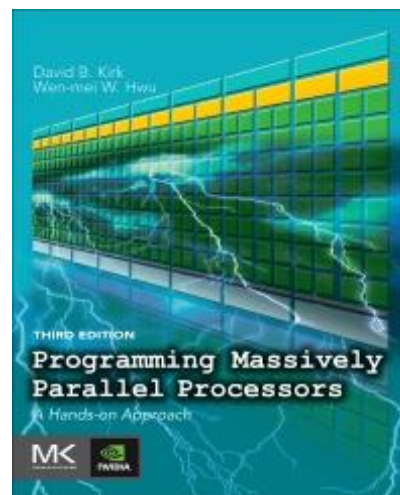
参考书：CUDA编程



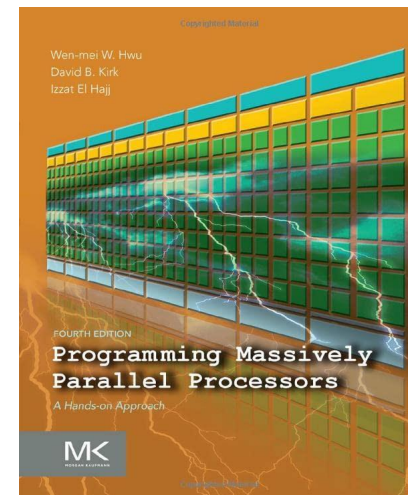
2010



2012



2017



2022

✓教材：Third Edition:

<https://www.sciencedirect.com/book/9780128119860/programming-massively-parallel-processors>

Contents

Preface.....	xv
Acknowledgements	xxi
CHAPTER 1 Introduction.....	1
1.1 Heterogeneous Parallel Computing	2
1.2 Architecture of a Modern GPU.....	6
1.3 Why More Speed or Parallelism?	8
1.4 Speeding Up Real Applications	10
1.5 Challenges in Parallel Programming	12
1.6 Parallel Programming Languages and Models	12
1.7 Overarching Goals	14
1.8 Organization of the Book.....	15
References	18
CHAPTER 2 Data Parallel Computing.....	19
2.1 Data Parallelism.....	20
2.2 CUDA C Program Structure	22
2.3 A Vector Addition Kernel	25
2.4 Device Global Memory and Data Transfer.....	27
2.5 Kernel Functions and Threading.....	32
2.6 Kernel Launch.....	37
2.7 Summary	38
Function Declarations.....	38
Kernel Launch.....	38
Built-in (Predefined) Variables	39
Run-time API.....	39
2.8 Exercises	39
References	41
CHAPTER 3 Scalable Parallel Execution.....	43
3.1 CUDA Thread Organization	43
3.2 Mapping Threads to Multidimensional Data	47
3.3 Image Blur: A More Complex Kernel	54
3.4 Synchronization and Transparent Scalability	58
3.5 Resource Assignment.....	60
3.6 Querying Device Properties.....	61
3.7 Thread Scheduling and Latency Tolerance.....	64
3.8 Summary	67
3.9 Exercises	67

CHAPTER 4 Memory and Data Locality	71
4.1 Importance of Memory Access Efficiency.....	72
4.2 Matrix Multiplication.....	73
4.3 CUDA Memory Types	77
4.4 Tiling for Reduced Memory Traffic.....	84
4.5 A Tiled Matrix Multiplication Kernel.....	90
4.6 Boundary Checks.....	94
4.7 Memory as a Limiting Factor to Parallelism	97
4.8 Summary	99
4.9 Exercises	100
CHAPTER 5 Performance Considerations.....	103
5.1 Global Memory Bandwidth	104
5.2 More on Memory Parallelism	112
5.3 Warps and SIMD Hardware.....	117
5.4 Dynamic Partitioning of Resources	125
5.5 Thread Granularity.....	127
5.6 Summary	128
5.7 Exercises	128
References	130
CHAPTER 6 Numerical Considerations	131
6.1 Floating-Point Data Representation.....	132
Normalized Representation of M	132
Excess Encoding of E	133
6.2 Representable Numbers	134
6.3 Special Bit Patterns and Precision in IEEE Format.....	138
6.4 Arithmetic Accuracy and Rounding	139
6.5 Algorithm Considerations.....	140
6.6 Linear Solvers and Numerical Stability.....	142
6.7 Summary	146
6.8 Exercises	147
References	147
CHAPTER 7 Parallel Patterns: Convolution	149
7.1 Background.....	150
7.2 1D Parallel Convolution—A Basic Algorithm	153
7.3 Constant Memory and Caching	156
7.4 Tiled 1D Convolution with Halo Cells	160
7.5 A Simpler Tiled 1D Convolution—General Caching	165
7.6 Tiled 2D Convolution with Halo Cells	166

7.7 Summary	172
7.8 Exercises	173
CHAPTER 8 Parallel Patterns: Prefix Sum.....	175
8.1 Background.....	176
8.2 A Simple Parallel Scan	177
8.3 Speed and Work Efficiency.....	181
8.4 A More Work-Efficient Parallel Scan.....	183
8.5 An Even More Work-Efficient Parallel Scan.....	187
8.6 Hierarchical Parallel Scan for Arbitrary-Length Inputs.....	189
8.7 Single-Pass Scan for Memory Access Efficiency.....	192
8.8 Summary	195
8.9 Exercises	195
References	196
CHAPTER 9 Parallel Patterns—Parallel Histogram Computation..	199
9.1 Background.....	200
9.2 Use of Atomic Operations	202
9.3 Block versus Interleaved Partitioning.....	206
9.4 Latency versus Throughput of Atomic Operations.....	207
9.5 Atomic Operation in Cache Memory	210
9.6 Privatization	210
9.7 Aggregation	211
9.8 Summary	213
9.9 Exercises	213
Reference.....	214
CHAPTER 10 Parallel Patterns: Sparse Matrix Computation	215
10.1 Background.....	216
10.2 Parallel SpMV Using CSR.....	219
10.3 Padding and Transposition.....	221
10.4 Using a Hybrid Approach to Regulate Padding.....	224
10.5 Sorting and Partitioning for Regularization.....	227
10.6 Summary	229
10.7 Exercises	229
References	230
CHAPTER 11 Parallel Patterns: Merge Sort.....	231
11.1 Background.....	231
11.2 A Sequential Merge Algorithm.....	233
11.3 A Parallelization Approach.....	234
11.4 Co-Rank Function Implementation.....	236

11.5 A Basic Parallel Merge Kernel	241
11.6 A Tiled Merge Kernel	242
11.7 A Circular-Buffer Merge Kernel.....	249
11.8 Summary	256
11.9 Exercises	256
Reference.....	256
CHAPTER 12 Parallel Patterns: Graph Search.....	257
12.1 Background.....	258
12.2 Breadth-First Search	260
12.3 A Sequential BFS Function	262
12.4 A Parallel BFS Function	265
12.5 Optimizations.....	270
Memory Bandwidth.....	270
Hierarchical Queues	271
Kernel Launch Overhead.....	272
Load Balance.....	273
12.6 Summary	273
12.7 Exercises	273
References	274
CHAPTER 13 CUDA Dynamic Parallelism.....	275
13.1 Background.....	276
13.2 Dynamic Parallelism Overview	278
13.3 A Simple Example	279
13.4 Memory Data Visibility.....	281
Global Memory	281
Zero-Copy Memory.....	282
Constant Memory	282
Local Memory	282
Shared Memory	283
Texture Memory	283
13.5 Configurations and Memory Management	283
Launch Environment Configuration	283
Memory Allocation and Lifetime	283
Nesting Depth.....	284
Pending Launch Pool Configuration	284
Errors and Launch Failures.....	284
13.6 Synchronization, Streams, and Events.....	285
Synchronization.....	285
Synchronization Depth	285
Streams	286
Events	287

13.7 A More Complex Example	287
Linear Bezier Curves	288
Quadratic Bezier Curves	288
Bezier Curve Calculation (Without Dynamic Parallelism)	288
Bezier Curve Calculation (With Dynamic Parallelism)	290
Launch Pool Size	292
Streams	292
13.8 A Recursive Example	293
13.9 Summary	297
13.10 Exercises	299
References	301
A13.1 Code Appendix	301

CHAPTER 14 Application Case Study—non-Cartesian Magnetic Resonance Imaging.....305

14.1 Background	306
14.2 Iterative Reconstruction	308
14.3 Computing F ^{HD}	310
Step 1: Determine the Kernel Parallelism Structure	312
Step 2: Getting Around the Memory Bandwidth Limitation	317
Step 3: Using Hardware Trigonometry Functions	323
Step 4: Experimental Performance Tuning	326
14.4 Final Evaluation	327
14.5 Exercises	328
References	329

CHAPTER 15 Application Case Study—Molecular Visualization and Analysis.....331

15.1 Background	332
15.2 A Simple Kernel Implementation	333
15.3 Thread Granularity Adjustment	337
15.4 Memory Coalescing	338
15.5 Summary	342
15.6 Exercises	343
References	344

CHAPTER 16 Application Case Study—Machine Learning.....345

16.1 Background	346
16.2 Convolutional Neural Networks	347
ConvNets: Basic Layers	348
ConvNets: Backpropagation	351
16.3 Convolutional Layer: A Basic CUDA Implementation of Forward Propagation	355

16.4 Reduction of Convolutional Layer to Matrix Multiplication	359
16.5 cuDNN Library	364
16.6 Exercises	366
References	367

CHAPTER 17 Parallel Programming and Computational Thinking.....369

17.1 Goals of Parallel Computing	370
17.2 Problem Decomposition	371
17.3 Algorithm Selection	374
17.4 Computational Thinking	379
17.5 Single Program, Multiple Data, Shared Memory and Locality	380
17.6 Strategies for Computational Thinking	382
17.7 A Hypothetical Example: Sodium Map of the Brain	383
17.8 Summary	386
17.9 Exercises	386
References	386

CHAPTER 18 Programming a Heterogeneous Computing Cluster.....387

18.1 Background	388
18.2 A Running Example	388
18.3 Message Passing Interface Basics	391
18.4 Message Passing Interface Point-to-Point Communication	393
18.5 Overlapping Computation and Communication	400
18.6 Message Passing Interface Collective Communication	408
18.7 CUDA-Aware Message Passing Interface	409
18.8 Summary	410
18.9 Exercises	410
Reference	411

CHAPTER 19 Parallel Programming with OpenACC.....413

19.1 The OpenACC Execution Model	414
19.2 OpenACC Directive Format	416
19.3 OpenACC by Example	418
The OpenACC Kernels Directive	419
The OpenACC Parallel Directive	422
Comparison of Kernels and Parallel Directives	424
OpenACC Data Directives	425
OpenACC Loop Optimizations	430
OpenACC Routine Directive	432
Asynchronous Computation and Data	434

19.4	Comparing OpenACC and CUDA.....	435
	Portability	435
	Performance.....	436
	Simplicity	436
19.5	Interoperability with CUDA and Libraries	437
	Calling CUDA or Libraries with OpenACC Arrays.....	437
	Using CUDA Pointers in OpenACC	438
	Calling CUDA Device Kernels from OpenACC.....	439
19.6	The Future of OpenACC.....	440
19.7	Exercises	441
CHAPTER 20 More on CUDA and Graphics Processing Unit Computing.....		443
20.1	Model of Host/Device Interaction.....	444
20.2	Kernel Execution Control	449
20.3	Memory Bandwidth and Compute Throughput.....	451
20.4	Programming Environment.....	453
20.5	Future Outlook	455
	References	456
CHAPTER 21 Conclusion and Outlook.....		457
21.1	Goals Revisited	457
21.2	Future Outlook	458
Appendix A: An Introduction to OpenCL.....		461
Appendix B: THRUST: a Productivity-oriented Library for CUDA.....		475
Appendix C: CUDA Fortran		493
Appendix D: An introduction to C++ AMP.....		515
Index		535

总结

