1. Record Nr. UNISA990001565520203316

Autore	FINE, Ben
Titolo	Il Capitale di Marx / Ben Fine ; [traduzione di Adriano Nardi]
Pubbl/distr/stampa	Napoli : Liguori, 1981
Edizione	[2. ed]
Descrizione fisica	91 p. ; 21 cm
Collana	Studi d'economia ; 2
Disciplina	330.01
Collocazione	II.1.D. 3264(XV F Coll 3/2)
Lingua di pubblicazione	Italiano
Formato	Materiale a stampa
Livello bibliografico	Monografia

2. Record Nr. UNINA9910453700903321

Autore	Suh Jung W
Titolo	Accelerating MATLAB with GPU computing : a primer with examples / / Jung W. Suh, Youngmin Kim
Pubbl/distr/stampa	Waltham, MA : , : Morgan Kaufmann, , 2014
ISBN	0-12-407916-4
Edizione	[First edition.]
Descrizione fisica	1 online resource (259 p.)
Altri autori (Persone)	KimYoungmin
Disciplina	518.0285
Soggetti	Graphics processing units - Programming Electronic books.
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Description based upon print version of record.
Nota di bibliografia	Includes bibliographical references and index.
Nota di contenuto	Front Cover; Accelerating MATLAB with GPU Computing; Copyright Page; Contents; Preface; Target Readers and Contents; Directions of this Book; GPU Utilization Using c-mex Versus Parallel Computing

Toolbox; Tutorial Approach Versus Case Study Approach; CUDA Versus OpenCL; 1 Accelerating MATLAB without GPU; 1.1 Chapter Objectives; 1.2 Vectorization; 1.2.1 Elementwise Operation; 1.2.2 Vector/Matrix Operation; 1.2.3 Useful Tricks; 1.3 Preallocation; 1.4 For-Loop; 1.5 Consider a Sparse Matrix Form; 1.6 Miscellaneous Tips; 1.6.1 Minimize File Read/Write Within the Loop
1.6.2 Minimize Dynamically Changing the Path and Changing the Variable Class 1.6.3 Maintain a Balance Between the Code Readability and Optimization; 1.7 Examples; 2 Configurations for MATLAB and CUDA; 2.1 Chapter Objectives; 2.2 MATLAB Configuration for c-mex Programming; 2.2.1 Checklists; 2.2.1.1 C/C++ Compilers; 2.2.1.2 NVIDIA CUDA Compiler nvcc; 2.2.2 Compiler Selection; 2.3 "Hello, mex!" using C-MEX; 2.3.1.1 Summary; 2.4 CUDA Configuration for MATLAB; 2.4.1 Preparing CUDA Settings; 2.5 Example: Simple Vector Addition Using CUDA; 2.5.1.1 Summary; 2.6 Example with Image Convolution
2.6.1 Convolution in MATLAB 2.6.2 Convolution in Custom c-mex; 2.6.3 Convolution in Custom c-mex with CUDA; 2.6.4 Brief Time Performance Profiling; 2.7 Summary; 3 Optimization Planning through Profiling; 3.1 Chapter Objectives; 3.2 MATLAB Code Profiling to Find Bottlenecks; 3.2.1 More Accurate Profiling with Multiple CPU Cores; 3.3 c-mex Code Profiling for CUDA; 3.3.1 CUDA Profiling Using Visual Studio; 3.3.2 CUDA Profiling Using NVIDIA Visual Profiler; 3.4 Environment Setting for the c-mex Debugger; 4 CUDA Coding with c-mex; 4.1 Chapter Objectives; 4.2 Memory Layout for c-mex
4.2.1 Column-Major Order 4.2.2 Row-Major Order; 4.2.3 Memory Layout for Complex Numbers in c-mex; 4.3 Logical Programming Model; 4.3.1 Logical Grouping 1; 4.3.2 Logical Grouping 2; 4.3.3 Logical Grouping 3; 4.4 Tidbits of GPU; 4.4.1 Data Parallelism; 4.4.2 Streaming Processor; 4.4.3 Steaming Multiprocessor; 4.4.4 Warp; 4.4.5 Memory; 4.5 Analyzing Our First Naive Approach; 4.5.1 Optimization A: Thread Blocks; 4.5.2 Optimization B; 4.5.3 Conclusion; 5 MATLAB and Parallel Computing Toolbox; 5.1 Chapter Objectives; 5.2 GPU Processing for Built-in MATLAB Functions; 5.2.1 Pitfalls in GPU Processing
5.3 GPU Processing for Non-Built-in MATLAB Functions 5.4 Parallel Task Processing; 5.4.1 MATLAB Worker; 5.4.2 parfor; 5.5 Parallel Data Processing; 5.5.1 spmd; 5.5.2 Distributed and Codistributed Arrays; 5.5.3 Workers with Multiple GPUs; 5.6 Direct use of CUDA Files without c-mex; 6 Using CUDA-Accelerated Libraries; 6.1 Chapter Objectives; 6.2 CUBLAS; 6.2.1 CUBLAS Functions; 6.2.2 CUBLAS Matrix-by-Matrix Multiplication; 6.2.2.1 Step 1; 6.2.2.2 Step 2; 6.2.2.3 Step 3; 6.2.2.4 Step 4; 6.2.2.5 Step 5; 6.2.2.6 Step 6; 6.2.2.7 Step 7; 6.2.2.8 Step 8; 6.2.2.9 Step 9
6.2.3 CUBLAS with Visual Profiler

Sommario/riassunto

Beyond simulation and algorithm development, many developers increasingly use MATLAB even for product deployment in computationally heavy fields. This often demands that MATLAB codes run faster by leveraging the distributed parallelism of Graphics Processing Units (GPUs). While MATLAB successfully provides high-level functions as a simulation tool for rapid prototyping, the underlying details and knowledge needed for utilizing GPUs make MATLAB users hesitate to step into it. Accelerating MATLAB with GPUs offers a primer on bridging this gap. Starting with the basics, setting