



ACCELERATING MATLAB WITH GPU COMPUTING

A Primer with Examples

MK
MORGAN KAUFMANN

Jung W. Suh
Youngmin Kim

Accelerating MATLAB with GPU Computing

This page intentionally left blank

Accelerating MATLAB with GPU Computing

A Primer with Examples

Jung W. Suh
Youngmin Kim



AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO
Morgan Kaufmann is an imprint of Elsevier



Acquiring Editor: Todd Green
Editorial Project Manager: Lindsay Lawrence
Project Manager: Mohana Natarajan
Designer: Matthew Limbert

Morgan Kaufmann is an imprint of Elsevier
225 Wyman Street, Waltham, MA 02451, USA
First edition 2014

Copyright © 2014 Elsevier Inc. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means electronic, mechanical, photocopying, recording or otherwise without the prior written permission of the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone (+44) (0) 1865 843830; fax (+44) (0) 1865 853333; email: permissions@elsevier.com. Alternatively you can submit your request online by visiting the Elsevier web site at <http://elsevier.com/locate/permissions>, and selecting Obtaining permission to use Elsevier material.

Notice

No responsibility is assumed by the publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein. Because of rapid advances in the medical sciences, in particular, independent verification of diagnoses and drug dosages should be made.

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

Library of Congress Cataloging-in-Publication Data

Application Submitted

ISBN: 978-0-12-408080-5

For information on all MK publications
visit our web site at www.mkp.com

Printed and bound in USA

14 15 16 17 18 10 9 8 7 6 5 4 3 2 1



Working together
to grow libraries in
developing countries

www.elsevier.com • www.bookaid.org

Contents

Preface	ix
1 Accelerating MATLAB without GPU	1
1.1 Chapter Objectives	1
1.2 Vectorization	1
1.2.1 Elementwise Operation	2
1.2.2 Vector/Matrix Operation	3
1.2.3 Useful Tricks	4
1.3 Preallocation	6
1.4 For-Loop	7
1.5 Consider a Sparse Matrix Form	7
1.6 Miscellaneous Tips	10
1.6.1 Minimize File Read/Write Within the Loop	10
1.6.2 Minimize Dynamically Changing the Path and Changing the Variable Class	10
1.6.3 Maintain a Balance Between the Code Readability and Optimization	10
1.7 Examples	10
2 Configurations for MATLAB and CUDA	19
2.1 Chapter Objectives	19
2.2 MATLAB Configuration for c-mex Programming	19
2.2.1 Checklists	19
2.2.2 Compiler Selection	20
2.3 “Hello, mex!” using C-MEX	22
2.4 CUDA Configuration for MATLAB	26
2.4.1 Preparing CUDA Settings	26
2.5 Example: Simple Vector Addition Using CUDA	28
2.6 Example with Image Convolution	33
2.6.1 Convolution in MATLAB	34
2.6.2 Convolution in Custom c-mex	35

2.6.3 Convolution in Custom <code>c-mex</code> with CUDA	38
2.6.4 Brief Time Performance Profiling	42
2.7 Summary	44
3 Optimization Planning through Profiling	45
3.1 Chapter Objectives	45
3.2 MATLAB Code Profiling to Find Bottlenecks	45
3.2.1 More Accurate Profiling with Multiple CPU Cores	49
3.3 <code>c-mex</code> Code Profiling for CUDA	52
3.3.1 CUDA Profiling Using Visual Studio	52
3.3.2 CUDA Profiling Using NVIDIA Visual Profiler	54
3.4 Environment Setting for the <code>c-mex</code> Debugger	65
4 CUDA Coding with <code>c-mex</code>	73
4.1 Chapter Objectives	73
4.2 Memory Layout for <code>c-mex</code>	73
4.2.1 Column-Major Order	73
4.2.2 Row-Major Order	76
4.2.3 Memory Layout for Complex Numbers in <code>c-mex</code>	77
4.3 Logical Programming Model	79
4.3.1 Logical Grouping 1	82
4.3.2 Logical Grouping 2	82
4.3.3 Logical Grouping 3	83
4.4 Tidbits of GPU	84
4.4.1 Data Parallelism	84
4.4.2 Streaming Processor	84
4.4.3 Streaming Multiprocessor	84
4.4.4 Warp	85
4.4.5 Memory	85
4.5 Analyzing Our First Naïve Approach	85
4.5.1 Optimization A: Thread Blocks	89
4.5.2 Optimization B	95
4.5.3 Conclusion	97
5 MATLAB and Parallel Computing Toolbox	99
5.1 Chapter Objectives	99
5.2 GPU Processing for Built-in MATLAB Functions	99
5.2.1 Pitfalls in GPU Processing	104

5.3 GPU Processing for Non-Built-in MATLAB Functions	106
5.4 Parallel Task Processing	108
5.4.1 MATLAB Worker	108
5.4.2 parfor	109
5.5 Parallel Data Processing	112
5.5.1 spmd	112
5.5.2 Distributed and Codistributed Arrays	116
5.5.3 Workers with Multiple GPUs	120
5.6 Direct use of CUDA Files without c-mex	120
6 Using CUDA-Accelerated Libraries	127
6.1 Chapter Objectives	127
6.2 CUBLAS	127
6.2.1 CUBLAS Functions	128
6.2.2 CUBLAS Matrix-by-Matrix Multiplication	128
6.2.3 CUBLAS with Visual Profiler	137
6.3 CUFFT	139
6.3.1 2D FFT with CUFFT	141
6.3.2 CUFFT with Visual Profiler	148
6.4 Thrust	151
6.4.1 Sorting with Thrust	151
6.4.2 Thrust with Visual Profiler	153
7 Example in Computer Graphics	157
7.1 Chapter Objectives	157
7.2 Marching Cubes	157
7.3 Implementation in MATLAB	161
7.3.1 Step 1	161
7.3.2 Step 2	163
7.3.3 Step 3	163
7.3.4 Step 4	164
7.3.5 Step 5	164
7.3.6 Step 6	165
7.3.7 Step 7	166
7.3.8 Step 8	167
7.3.9 Step 9	167
7.3.10 Time Profiling	174

7.4	Implementation in c-mex with CUDA	175
7.4.1	Step 1	175
7.4.2	Step 2	178
7.4.3	Time Profiling	179
7.5	Implementation Using c-mex and GPU	180
7.5.1	Step 1	180
7.5.2	Step 2	182
7.5.3	Step 3	182
7.5.4	Step 4	188
7.5.5	Step 5	188
7.5.6	Time Profiling	189
7.6	Conclusion	190
8	CUDA Conversion Example: 3D Image Processing	193
8.1	Chapter Objectives	193
8.2	MATLAB Code for Atlas-Based Segmentation	193
8.2.1	Background of Atlas-Based Segmentation	193
8.2.2	MATLAB Codes for Segmentation	194
8.3	Planning for CUDA Optimization Through Profiling	203
8.3.1	Profiling MATLAB Code	203
8.3.2	Analyze the Profiling Results and Planning CUDA Optimization	207
8.4	CUDA Conversion 1 - Regularization	210
8.5	CUDA Conversion 2 - Image Registration	215
8.6	CUDA Conversion Results	228
8.7	Conclusion	228
	Appendix 1: Download and Install the CUDA Library	233
	Appendix 2: Installing NVIDIA Nsight into Visual Studio	239
	Bibliography	243
	Index	245

Preface

MATLAB is a widely used simulation tool for rapid prototyping and algorithm development. Many laboratories and research institutions face growing demands to run their MATLAB codes faster for computationally heavy projects after simple simulations. Since MATLAB uses a vector/matrix representation of data, which is suitable for parallel processing, it can benefit a lot from GPU acceleration.

Target Readers and Contents

This book is aimed primarily at the graduate students and researchers in the field of engineering, science, and technology who need huge data processing without losing the many benefits of MATLAB. However, MATLAB users come from various backgrounds and do not necessarily have much programming experience. For those whose backgrounds are not from programming, GPU acceleration for MATLAB may distract their algorithm development and introduce unnecessary hassles, even when setting the environment. This book targets the readers who have some or a lot of experience on MATLAB coding but not enough depth in either C coding or the computer architecture for parallelization. So readers can focus more on their research and work by avoiding non-algorithmic hassles in using GPU and CUDA in MATLAB.

As a primer, the book will start with the basics, walking through the process of setting MATLAB for CUDA (in Windows and Mac OSX), creating `c-mex` and `m-file` profiling, then guide the users through the expert-level topics such as third-party CUDA libraries. It also provides many practical ways to modify users' MATLAB codes to better utilize the immense computational power of graphics processors.

This book guides the reader to dramatically maximize the MATLAB speed using NVIDIA's Graphics Processing Unit (GPU). NVIDIA's Compute Unified Device Architecture (CUDA) is a parallel computing architecture originally designed for computer games but is getting a reputation in the general science and technology fields for its efficient massive computation power. From this book, the reader can take advantage of the parallel processing power of GPU and abundant CUDA scientific libraries for accelerating MATLAB code with no or less effort and time, and bring readers' researches and works to a higher level.

Directions of this Book

GPU Utilization Using c-mex Versus Parallel Computing Toolbox

This book deals with Mathworks's Parallel Computing Toolbox in Chapter 5. Although Mathworks's Parallel Computing Toolbox is a useful tool for speeding