

# Distributed Graph Analytics

Unnikrishnan Cheramangalath • Rupesh Nasre •  
Y. N. Srikant

# Distributed Graph Analytics

Programming, Languages,  
and Their Compilation

Unnikrishnan Cheramangalath  
Department of Computer Science  
and Engineering  
Indian Institute of Technology Palakkad  
Palakkad, Kerala, India

Rupesh Nasre  
Department of Computer Science  
and Engineering  
Indian Institute of Technology Madras  
Chennai, India

Y. N. Srikant  
Department of Computer Science  
and Automation  
Indian Institute of Science  
Bangalore, India

ISBN 978-3-030-41885-4      ISBN 978-3-030-41886-1 (eBook)  
<https://doi.org/10.1007/978-3-030-41886-1>

© Springer Nature Switzerland AG 2020

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG.  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

Graphs model several real-world phenomena. Graph algorithms have been an interesting area of study and research due to rich structural information embedded into graphs. However, it is only in the last couple of decades that systemic aspects of graph algorithms have been pursued more prominently. Especially with the advent of large networks, and storage of huge amount of unstructured data, efficient graph processing has become a crucial piece in the performance puzzle of the underlying applications. For example, efficient graph processing is central to detection of online frauds, improvements in advertising and business, study of how diseases spread, computational geometry, and natural language processing, to name a few.

In parallel, we witnessed prominent innovations in computer architecture. Shared memory systems with a large number of cores, NUMA architecture, accelerators such as GPUs and FPGAs, and large decentralized systems are so common that high-performance computing is no longer a niche area. Almost every field, be it physics, biology, or material science, need to depend on such architectures for efficient simulation of a natural phenomenon. On the industry front, data warehouses are becoming bulky with pressures to store, maintain, and analyze tremendous amount of structured and unstructured data. Graph databases, textgraphs, and their modeling as hypergraphs pose new challenges. Processing such multi-modal data requires different algorithms. In addition, an algorithm needs to be tuned for a certain backend hardware for efficient implementation, because an optimized implementation for one hardware may not be efficient on the other.

Our goal in this book is to marry these two important trends: graph algorithms and high performance computing. Efficient and scalable execution of graph processing applications demands innovations at multiple levels: algorithm, its associated data structures, their implementation, and their implementation tuned to a certain hardware. Managing complexity at so many levels of abstraction is clearly a challenge for users. The issue gets exacerbated as domain experts may not be HPC experts, and vice versa. Therefore, the role of programming languages and the associated compilers is crucial to automate efficient code generation for various architectures. This book pens down essentials on these aspects.

We divide this book in three parts: programming, languages, and their compilation. The first part deals with manual parallelization of graph algorithms. It uncovers various parallelization patterns we encounter while dealing especially with graphs. The second part exploits these patterns to provide language constructs using which a graph algorithm can be specified. A programmer can work only with those language constructs, without worrying about their implementation. The implementation, which is the heart of the third part, is handled by a compiler, which can specialize code generation for a backend device. We present a few suggestive results on different platforms, which justify the theory and practice covered in the book. Together, the three parts provide the essential ingredients in creating a high-performing graph application.

While this book is pitched at a graduate or advanced undergraduate level as a specialized elective in universities, to make the matter accessible, we have also included a brief background on elementary graph algorithms, parallel computing, and GPUs. It is possible to read most of the chapters independently, if the readers are familiar with the basics of parallel programming and graph algorithms. To highlight recent advances, we also discuss dynamic graph algorithms, which pose new challenges at the algorithmic and language levels, as well as for code generation. To make the discussion more concrete, we use a case study using Falcon, a domain-specific language for graph algorithms. The book ends with a section on future directions which contains several pointers to topics that seem promising for future research.

We believe this book provides you wings to explore the exciting area of distributed graph analytics with zeal, and that it would help practitioners scale their graph algorithms to newer heights.

We would like to especially thank Ralf Gerstner, Executive Editor of Computer Science at Springer-Verlag for his encouragement and patient advice.

Palakkad, India  
Chennai, India  
Bangalore, India  
January 2020

Unnikrishnan Cheramangalath  
Rupesh Nasre  
Y. N. Srikant

# Contents

- 1 Introduction to Graph Analytics ..... 1**
  - 1.1 What Is Graph Analytics? ..... 1
  - 1.2 Graph Preliminaries ..... 3
  - 1.3 Graph Storage Formats ..... 6
    - 1.3.1 Adjacency Matrix ..... 6
    - 1.3.2 Compressed Sparse Row ..... 6
    - 1.3.3 Incidence Matrix ..... 7
    - 1.3.4 Other Formats ..... 7
  - 1.4 Graph Partitioning Strategies ..... 8
  - 1.5 Heterogeneous Distributed Systems ..... 10
  - 1.6 Programming Libraries and APIs ..... 11
    - 1.6.1 OpenMP ..... 11
    - 1.6.2 CUDA ..... 13
    - 1.6.3 OpenCL ..... 14
    - 1.6.4 Thrust..... 15
    - 1.6.5 MPI ..... 15
  - 1.7 Graph Analytics in Real-World Applications..... 20
  - 1.8 Graph Analytics Challenges ..... 20
    - 1.8.1 Classification of Graphs ..... 20
    - 1.8.2 Irregular Computation ..... 21
    - 1.8.3 Heterogeneous Hardware..... 22
    - 1.8.4 Distributed Execution..... 22
    - 1.8.5 Atomic Operations and Synchronization ..... 23
    - 1.8.6 Challenges in Programming ..... 24
  - 1.9 Programming Abstractions for Graph Analytics ..... 24
    - 1.9.1 Frameworks ..... 24
    - 1.9.2 Domain Specific Languages..... 27

<b>2</b>	<b>Graph Algorithms and Applications</b>	31
2.1	Introduction	31
2.2	Fundamental Graph Algorithms	33
2.2.1	Breadth-First Search (BFS)	34
2.2.2	Depth-First Search	36
2.2.3	Single Source Shortest Path (SSSP)	37
2.2.4	All Pairs Shortest Path Computation (APSP)	40
2.2.5	Strongly Connected Components (SCC)	43
2.2.6	Weakly Connected Components (WCC)	46
2.2.7	Minimum Spanning Tree (MST)	46
2.2.8	Maximal Independent Set (MIS) Computation	48
2.3	Other Algorithms	49
2.3.1	Pagerank Algorithm	49
2.3.2	Triangle Counting	50
2.3.3	Graph Coloring	51
2.3.4	K-Core	52
2.3.5	Betweenness Centrality (BC)	53
2.4	Applications of Graph Analytics in Different Domains	58
2.4.1	Graph Mining	58
2.4.2	Graph Databases (NoSQL)	58
2.4.3	Text Graphs in Natural Language Processing	59
<b>3</b>	<b>Efficient Parallel Implementation of Graph Algorithms</b>	61
3.1	Introduction	61
3.2	Issues in Programming Parallel Graph Algorithms	62
3.2.1	Parallelism	62
3.2.2	Atomicity	62
3.2.3	An Example	63
3.2.4	Graph Topology and Classification	64
3.2.5	Push vs. Pull Computation	65
3.2.6	Topology Driven vs. Data Driven	67
3.3	Different Ways of Solving a Problem	69
3.3.1	Vertex-Based SSSP Computation	71
3.3.2	Worklist-Based SSSP Computation	73
3.3.3	Edge-Based SSSP Computation	73
3.3.4	$\Delta$ -Stepping SSSP	74
3.4	A Note on Efficient Parallel Implementations of Graph Algorithms	76
3.5	Some Important Parallel Graph Algorithms	77
3.5.1	Parallel Computation of Maximal Independent Sets	77
3.5.2	Parallel Computation of Strongly Connected Components	78
3.5.3	Parallel Computation of Connected Components	79
3.5.4	Parallel Computation of Betweenness Centrality (BC)	83
3.5.5	Parallel Union-Find and Applications	88

3.6	Graph Analytics on Distributed Systems .....	93
3.6.1	Distributed System with CPUs .....	96
3.6.2	Distributed System with CPUs and GPUs .....	96
3.6.3	Multi-GPU Machine .....	97
<b>4</b>	<b>Graph Analytics Frameworks .....</b>	<b>99</b>
4.1	Introduction .....	99
4.2	Frameworks—Merits and Demerits .....	100
4.3	Models for Graph Analytics .....	101
4.3.1	Bulk Synchronous Parallel (BSP) .....	101
4.3.2	Map-Reduce .....	105
4.3.3	Asynchronous Execution .....	105
4.3.4	External Memory Frameworks .....	107
4.3.5	Gather-Apply-Scatter Model .....	107
4.3.6	Inspector-Executor .....	108
4.3.7	Advance-Filter-Compute .....	110
4.4	Frameworks for Single Machines .....	111
4.4.1	Multi-Core CPU .....	113
4.4.2	Single GPU .....	113
4.4.3	Multi GPU .....	115
4.5	Frameworks for Distributed Systems .....	119
4.5.1	Distributed System with CPU .....	120
4.5.2	Distributed System with CPU and GPU .....	121
<b>5</b>	<b>GPU Architecture and Programming Challenges .....</b>	<b>123</b>
5.1	Introduction .....	123
5.2	Graphics Processing Unit (GPU) .....	124
5.2.1	GPU Architecture .....	124
5.2.2	Multi-GPU Machine .....	125
5.2.3	Distributed GPU Systems .....	127
5.3	General Purpose Computing on GPU (GPGPU) .....	127
5.3.1	CUDA Programming .....	127
5.3.2	GPU Thread Scheduling .....	128
5.3.3	Warp Based Execution .....	130
5.3.4	Memory Hierarchy .....	132
5.4	Graph Analytics on GPU .....	133
5.4.1	Topology Driven Algorithms .....	134
5.4.2	Other Elementary Graph Algorithms .....	136
<b>6</b>	<b>Dynamic Graph Algorithms .....</b>	<b>137</b>
6.1	Introduction .....	137
6.2	Dynamic Algorithms for Elementary Graph Problems .....	139
6.3	Dynamic Shortest Path Computation .....	139
6.3.1	Incremental Dynamic SDSP Computation .....	140
6.3.2	Decremental Dynamic SDSP Computation .....	144



6.4	Computational Geometry Algorithms.....	146
6.4.1	Delaunay Triangulation (DT) .....	146
6.4.2	Delaunay Mesh Refinement (DMR) .....	146
6.4.3	Parallel Delaunay Mesh Refinement (DMR) .....	148
6.5	Challenges in Implementing Dynamic Algorithms.....	149
6.5.1	Dynamic Memory Management .....	150
6.5.2	Parallel Computation .....	150
<b>7</b>	<b>Falcon: A Domain Specific Language for Graph Analytics .....</b>	<b>153</b>
7.1	Introduction .....	153
7.2	Overview .....	154
7.2.1	SSSP Program in Falcon .....	155
7.3	Falcon Data Types .....	158
7.4	Falcon API .....	159
7.5	Constructs for Parallel Execution and Synchronization .....	160
7.5.1	section and parallel sections Statements.....	162
7.6	Code Generation—Single Machine .....	162
7.6.1	Compilation of Data Types .....	165
7.6.2	Extra Property Memory Allocation .....	165
7.6.3	Compilation of foreach Statement .....	165
7.6.4	Data Transfer Between CPU and GPU .....	166
7.6.5	Compiling single Statement.....	167
7.6.6	Compiling Parallel Sections .....	168
7.6.7	Handling Reduction Operations.....	168
7.7	Falcon for Distributed Systems .....	170
7.7.1	Storage of Subgraphs in Falcon .....	170
7.7.2	Initialization Code .....	171
7.7.3	Allocation and Synchronization of Global Variables .....	172
7.7.4	Distributed Communication and Synchronization .....	173
7.7.5	Code Generation for Set and Collection Data Types.....	175
7.7.6	Code Generation for Parallel and Synchronization Statements .....	176
7.7.7	foreach Statement .....	178
<b>8</b>	<b>Experiments, Evaluation and Future Directions .....</b>	<b>181</b>
8.1	Introduction .....	181
8.2	Single Machine Experiments .....	181
8.2.1	Random and R-MAT Graphs .....	184
8.2.2	Road Network Graphs .....	184
8.2.3	Connected Components and MST .....	187
8.2.4	Summary .....	188
8.3	Distributed Systems .....	188
8.3.1	Multi-GPU .....	189
8.3.2	GPU Cluster.....	190
8.3.3	CPU Cluster.....	190

- 8.3.4 CPU + CPU Cluster ..... 190
    - 8.3.5 Results..... 191
  - 8.4 Future Directions in Distributed Graph Analytics ..... 191
    - 8.4.1 Custom Graph Processing ..... 191
    - 8.4.2 Adaptive Graph Partitioning and Processing ..... 192
    - 8.4.3 Hardware for Graph Processing ..... 192
    - 8.4.4 Handling Dynamic Updates ..... 193
    - 8.4.5 Performance Modeling ..... 193
    - 8.4.6 Learning to Aid Graph Processing ..... 193
- References** ..... 195
- Index** ..... 205