Copyright ©1991
Edoardo Stanley Biagioni
All Rights Reserved

to Jade

Abstract

Scan Directed Load Balancing

by
Edoardo Stanley Biagioni

Scan Directed Load Balancing is an algorithm for dynamic load balancing on k-dimensional mesh-connected computers. In one dimension, a scan computes the load to the left of each processor and the total load. The difference between the load to the left and the average load times the number of processors to the left is the number of active data points to be shifted to balance the load.

In two dimensions, scans are used to compute the load of each column of data and the load to the left of each column of processors. The difference between the load to the left and the average load times the number of processor columns to the left is the load of the data columns to be shifted by this column of processors; the computation is repeated for the rows. In more than two dimensions, the load is computed for (hyper-) planes of data and is balanced by shifting data (hyper-) planes among processor (hyper-) planes; the basic one-dimensional step of the algorithm is repeated k times.

The algorithm targets applications that can be expressed as local computations on a grid and whose dynamic activity is unevenly distributed. In one such computation, Variable Conductance Diffusion, different pixels diffuse for different numbers of iterations, with those that require many iterations unevenly distributed and, in the absence of load balancing, often located on a small subset of the processors. Scan Directed Load Balancing is effective in speeding up Variable Conductance Diffusion for a class of images (512×512 pixel), and very effective for selected images in which only a small fraction of the pixels requires many iterations. This implementation, and that for a parallel interpreter for the functional language FFP that uses Scan Directed Load Balancing to both manage storage and balance load, are for UNC's MasPar MP-1, with 4096 processors embedded in a 2-dimensional mesh.

Scan Directed Load Balancing does not guarantee optimal load balance, but always assigns neighboring data to the same or to neighboring processors, so communication between neighboring data is local and fast.

Acknowledgements

I very grateful to my entire dissertation committee, Vernon Chi, Gyula Magó, Jan Prins, Donald Stanat, and Jennifer Welch. Their continued dedication has contributed immensely to the quality of this dissertation (and to the time required for its completion); their comments and suggestions have been extremely useful and helpful, and have pointed out many interesting paths to pursue and many areas that needed further study for this work to be complete. Chapters 3 and 4 of this dissertation were originally a paper [11] co-authored by Jan Prins.

I am grateful to my family, Alessandra, Andrea, Carlo, Dianora, Ettore, Marcello, and Nancy, whose continued patience and support, both before and after I entered graduate school, has helped me focus and persevere in reaching this once distant goal.

I am grateful to all my friends, who have made life enjoyable while this work was going on. Those that must be mentioned include Jade Goldstein, without whom I might not have finished, J. Nievergelt, who first brought me to Chapel Hill, Will Partain, Bill Gibson, Klaus Hinrichs, Henry Eck, Satoru Ishii; countless more have helped me grow and get through this otherwise difficult and lonely time.

I want to thank the people of North Carolina for providing support and encouragement to me and my department, and the Office of Naval Research for supporting my work through grant #N00014-86-K-0680.

Contents

1 Introduction **1**

2 Background **6**

2.1 Load Balancing 6

2.1.1 Grain Size 7

2.1.2 Static and Dynamic Load Balancing 8

2.1.3 Load Sharing 11

2.1.4 Constrained Load Balancing 11

2.1.5 Computational Load for Subsets of Data 14

2.2 Scans 16

2.3 Continuum Computations 19

2.3.1 Definition and Examples 19

2.3.2 Variable Conductance Diffusion 21

2.3.3 Reduction of FFP 22

2.3.4 Data Distribution 26

2.4 Mesh Connected Computers 27

3 The Algorithm **31**

3.1 Assumptions and Definition of Terms 31

3.1.1 Domain Decomposition 31

3.1.2 Distance and Neighborhood 33

3.1.3 Locality Property 34

3.1.4 The Locality Property of OMS Decompositions 36

3.2 One Dimensional Load Balancing 37

3.2.1 Assumptions 37

3.2.2 Scans 38

3.2.3 Load Computation Step 39

3.2.4 Data Movement Step 41
3.3 Multi Dimensional Load Balancing 43
3.3.1 Assumptions 43
3.3.2 Reductions 48
3.3.3 Load Computation Step 49
3.3.4 Data Movement Step 52
3.4 Improvements 52
3.4.1 Buffer Overflows 53
3.4.2 Avoiding Buffer Overflows 55
3.4.3 Buffer Underflows 59
3.4.4 Efficient Management of Active Processes 61
3.4.5 Computing with Integers 64
3.5 Performance 66
3.5.1 Load Balance Achieved 66
3.5.2 Load Balance with Limited Buffers 67
3.5.3 Satisfying the Locality Property 68
3.5.4 Execution Time of Load Computation Step 70
3.5.5 Execution Time of Data Movement Step 73

4 Load Balancing for Variable Conductance Diffusion 75
4.1 Variable Conductance Diffusion 75
4.1.1 Background 75
4.1.2 Algorithm 76
4.2 Parallel Implementation 77
4.2.1 Decomposition and Communication 77
4.2.2 Data 78
4.2.3 Input Data 79
4.3 Load Balancing for Diffusion 79
4.3.1 Applicability for Load Balancing 79
4.3.2 Data Structures 82
4.3.3 Iteration Step 83
4.4 Measuring Performance 83
4.5 Results 85
4.5.1 Test Procedure 85
4.5.2 Average Performance 86
4.5.3 Optimal Performance 88
4.6 Analysis 92

4.6.1 Modeling the Load 92
4.6.2 Time Required to Balance Load 93
4.6.3 Modeling Total Time 96
4.6.4 Summary 97

5 MIMD on SIMD **99**
5.1 Motivation 99
5.1.1 Graph Reduction 99
5.1.2 Linear Reduction 101
5.1.3 MIMD and SIMD Computers 102
5.2 The FFP Machine 104
5.2.1 The FFP Language 104
5.2.2 Partitioning the FFP Machine 106
5.2.3 Summary of FFP 107
5.2.4 Sample Execution 110
5.3 The SOP Language 112
5.3.1 Objects 112
5.3.2 Expressions 114
5.3.3 Primitives 114
5.4 The SOP Interpreter 117
5.4.1 Programming Environment 117
5.4.2 Design 122
5.4.3 Memory Map 124
5.4.4 Storage Manager 126
5.4.5 Communication 127
5.5 Comparison of Implementations 130
5.5.1 Overview 130
5.5.2 Sequential Implementation 131
5.5.3 Connection Machine Implementations 132
5.5.4 MP-1 Implementations 133
5.5.5 Comparison and Analysis 134
5.5.6 Performance of Scan Directed Load Balancing 135
5.5.7 SOP Architectures 136

6 Summary and Conclusions **139**
6.1 Summary 139
6.2 Future Work 141

6.3 Conclusions 142

A Source Code for the SOP Environment **144**
A.1 `nonrec` 144
A.2 `tosop` 148
A.3 SOP Assembler 151

B FP Benchmarks **153**
B.1 Integer Sum 153
B.2 n Queens Problem 153
B.3 Primes 154

Bibliography **156**

List of Figures

2.1 Costs and Benefits of Dynamic Load Balancing 10
2.2 Additive and Maximal Loads for Planes of Data 16
2.3 Reducible FFP expressions and their reduced values 25

3.1 Orthogonal, Monotonic, Surjective Decomposition μ. 37
3.2 One-Dimensional Scan Directed Load Balancing 39
3.3 Example of values produced during the Load Computation Step. 40
3.4 Load Transfer in the Data Movement Step. 42
3.5 Explanation of Terms for $k = 2$ 45
3.6 Data and Processor hyperplanes 46
3.7 Multi Reductions on a $3 \times 3 \times 2$ Array 50
3.8 Scan Directed Load Balancing for More than One Dimension 51
3.9 Perfect load balance requires large memories. 54
3.10 One-Dimensional Load Balancing with Limited Buffers 55
3.11 Derivation of α in one Dimension 57
3.12 Non-integral Number of Points Assigned to a Processor . . . 59
3.13 Multi Dimensional Load Balancing with Limited Buffers . . . 60
3.14 One dimensional `put()` using doubly-linked lists 62
3.15 Example of Floating Point Arithmetic Causing Buffer Overflows 65

4.1 Algorithm for Variable Conductance Diffusion 77
4.2 The Original Image 80
4.3 The Image Produced by Variable Conductance Diffusion . . . 80
4.4 Optimized Algorithm for Variable Conductance Diffusion . . 81
4.5 Average Speedup given by Scan Directed Load Balancing . . 87
4.6 Average Speedup for 512×512 Images 89
4.7 Average Speedup by Balancing in Different Dimensions . . . 90
4.8 Speedup for image `bowahes8.im` 91

5.1 Linear Reduction of a Simple Expression 101
5.2 Factorial for FP and FFP 105
5.3 Simultaneous Computation of Independent Scans 107
5.4 Grammar for Structured Strings 113
5.5 Use of metacomposition to remove recursion from FFP 119
5.6 SOP Code for the FFP function `distr` 120
5.7 Sample Execution of `neg -3` on the SOP Interpreter 121
5.8 Parallel Row Right Scan 128
5.9 Algorithm for Right Scan on the MP-1 128

List of Tables

2.1 Functional Forms for vector processing in FFP 23

3.1 Notation 35
3.2 Notation for One Dimension 38
3.3 Notation for Multiple Dimensions 44

4.1 Estimate of Load Balancing time for `bowahes8.im` 94
4.2 Estimate of Load Balancing time for `im05.im` 96

5.1 Summary of SOPprimitives 115
5.2 Translators used with SOP 118
5.3 Memory Map for a SOP Cell 125
5.4 Time (seconds) for Benchmarks using `fpc` 131
5.5 Graphinator Times (seconds) for Integer Sum 133
5.6 Time (seconds) for Benchmarks using SOP and Risc 134