Table of Contents

Preface
Santosh Pande and Dharma P. Agrawal V

Introduction
Santosh Pande and Dharma P. Agrawal XXI

1 Compiling for Distributed Memory Multiprocessors XXI
1.1 Motivation XXI
1.2 Complexity XXII
1.3 Outline of the Monograph XXII
1.4 Future Directions XXVII

Section I : Languages

Chapter 1. High Performance Fortran 2.0
Ken Kennedy and Charles Koelbel 3

1 Introduction 3
2 History and Overview of HPF 3
3 Data Mapping 7
3.1 Basic Language Features 7
3.2 Advanced Topics 13
4 Data Parallelism 18
4.1 Basic Language Features 19
4.2 Advanced Topics 29
5 Task Parallelism 34
5.1 EXTRINSIC Procedures 34
5.2 The TASK_REGION Directive 37
6 Input and Output 39
7 Summary and Future Outlook 41

Chapter 2. The Sisal Project: Real World Functional Programming
Jean-Luc Gaudiot, Tom DeBoni, John Feo, Wim Böhm, Walid Najjar, and Patrick Miller ... 45

1 Introduction ... 45
2 The Sisal Language: A Short Tutorial ... 46
3 An Early Implementation: The Optimizing Sisal Compiler ... 49
3.1 Update in Place and Copy Elimination ... 49
3.2 Build in Place ... 50
3.3 Reference Counting Optimization ... 51
3.4 Vectorization ... 51
3.5 Loop Fusion, Double Buffering Pointer Swap, and Inversion ... 51
4 Sisal90 ... 53
4.1 The Foreign Language Interface ... 54
5 A Prototype Distributed-Memory SISAL Compiler ... 58
5.1 Base Compiler ... 59
5.2 Rectangular Arrays ... 59
5.3 Block Messages ... 60
5.4 Multiple Alignment ... 60
5.5 Results ... 61
5.6 Further Work ... 62
6 Architecture Support for Multithreaded Execution ... 62
6.1 Blocking and Non-blocking Models ... 63
6.2 Code Generation ... 64
6.3 Summary of Performance Results ... 68
7 Conclusions and Future Research ... 69

Chapter 3. HPC++ and the HPC++Lib Toolkit
Dennis Gannon, Peter Beckman, Elizabeth Johnson, Todd Green, and Mike Levine ... 73

1 Introduction ... 73
2 The HPC++ Programming and Execution Model ... 74
2.1 Level 1 HPC++ ... 75
2.2 The Parallel Standard Template Library ... 76
2.3 Parallel Iterators ... 77
2.4 Parallel Algorithms ... 77
2.5 Distributed Containers ... 78
3 A Simple Example: The Spanning Tree of a Graph ... 78
4 Multi-threaded Programming ... 82
4.1 Synchronization ... 84
4.2 Examples of Multi-threaded Computations ... 92
5 Implementing the HPC++ Parallel Loop Directives ... 96

6 Multi-context Programming and Global Pointers 99
6.1 Remote Function and Member Calls 101
6.2 Using Corba IDL to Generate Proxies 103
7 The SPMD Execution Model 105
7.1 Barrier Synchronization and Collective Operations 105
8 Conclusion 106

Chapter 4. A Concurrency Abstraction Model for Avoiding Inheritance Anomaly in Object-Oriented Programs
Sandeep Kumar and Dharma P. Agrawal 109

1 Introduction 109
2 Approaches to Parallelism Specification 113
2.1 Issues in Designing a COOPL 113
2.2 Issues in Designing Libraries 114
3 What Is the Inheritance Anomaly? 115
3.1 State Partitioning Anomaly (*SPA*) 116
3.2 History Sensitiveness of Acceptable States Anomaly (*HSASA*) . 118
3.3 State Modification Anomaly (*SMA*) 118
3.4 Anomaly A 119
3.5 Anomaly B 120
4 What Is the Reusability of Sequential Classes? 120
5 A Framework for Specifying Parallelism 121
6 Previous Approaches 122
7 The Concurrency Abstraction Model 123
8 The CORE Language 126
8.1 Specifying a Concurrent Region 126
8.2 Defining an AC 126
8.3 Defining a Parallel Block 127
8.4 Synchronization Schemes 129
9 Illustrations 129
9.1 Reusability of Sequential Classes 130
9.2 Avoiding the Inheritance Anomaly 131
10 The Implementation Approach 133
11 Conclusions and Future Directions 134

Section II : Analysis

Chapter 5. Loop Parallelization Algorithms
Alain Darte, Yves Robert, and Frédéric Vivien 141

1 Introduction 141
2 Input and Output of Parallelization Algorithms 142
2.1 Input: Dependence Graph 143
2.2 Output: Nested Loops 144

3 Dependence Abstractions . . . 145
3.1 Dependence Graphs and Distance Sets . . . 145
3.2 Polyhedral Reduced Dependence Graphs . . . 147
3.3 Definition and Simulation of Classical Dependence Representations . . . 148
4 Allen and Kennedy's Algorithm . . . 149
4.1 Algorithm . . . 150
4.2 Power and Limitations . . . 151
5 Wolf and Lam's Algorithm . . . 152
5.1 Purpose . . . 153
5.2 Theoretical Interpretation . . . 153
5.3 The General Algorithm . . . 154
5.4 Power and Limitations . . . 155
6 Darte and Vivien's Algorithm . . . 156
6.1 Another Algorithm Is Needed . . . 156
6.2 Polyhedral Dependences: A Motivating Example . . . 158
6.3 Illustrating Example . . . 160
6.4 Uniformization Step . . . 162
6.5 Scheduling Step . . . 162
6.6 Schematic Explanations . . . 165
6.7 Power and Limitations . . . 166
7 Feautrier's Algorithm . . . 167
8 Conclusion . . . 169

Chapter 6. Array Dataflow Analysis
Paul Feautrier . . . 173

1 Introduction . . . 173
2 Exact Array Dataflow Analysis . . . 176
2.1 Notations . . . 176
2.2 The Program Model . . . 176
2.3 Data Flow Analysis . . . 181
2.4 Summary of the Algorithm . . . 189
2.5 Related Work . . . 190
3 Approximate Array Dataflow Analysis . . . 190
3.1 From ADA to FADA . . . 191
3.2 Introducing Parameters . . . 195
3.3 Taking Properties of Parameters into Account . . . 197
3.4 Eliminating Parameters . . . 201
3.5 Related Work . . . 202
4 Analysis of Complex Statements . . . 204
4.1 What Is a Complex Statement . . . 204
4.2 ADA in the Presence of Complex Statements . . . 206
4.3 Procedure Calls as Complex Statements . . . 206

5 Applications of ADA and FADA 208
5.1 Program Comprehension and Debugging 209
5.2 Parallelization 211
5.3 Array Expansion and Array Privatization 212
6 Conclusions 214
A Appendix : Mathematical Tools 214
A.1 Polyhedra and Polytopes 214
A.2 Z-modules 215
A.3 Z-polyhedra 216
A.4 Parametric Problems 216

Chapter 7. Interprocedural Analysis Based on Guarded Array Regions
Zhiyuan Li, Junjie Gu, and Gyungho Lee 221

1 Introduction 221
2 Preliminary 223
2.1 Traditional Flow-Insensitive Summaries 223
2.2 Array Data Flow Summaries 225
3 Guarded Array Regions 226
3.1 Operations on GAR's 228
3.2 Predicate Operations 230
4 Constructing Summary GAR's Interprocedurally 232
4.1 Hierarchical Supergraph 232
4.2 Summary Algorithms 233
4.3 Expansions 235
5 Implementation Considerations 238
5.1 Symbolic Analysis 238
5.2 Region Numbering 239
5.3 Range Operations 240
6 Application to Array Privatization and Preliminary Experimental Results 240
6.1 Array Privatization 241
6.2 Preliminary Experimental Results 241
7 Related Works 243
8 Conclusion 244

Chapter 8. Automatic Array Privatization
Peng Tu and David Padua 247

1 Introduction 247
2 Background 248
3 Algorithm for Array Privatization 250
3.1 Data Flow Framework 250
3.2 Inner Loop Abstraction 252
3.3 An Example 256

3.4 Profitability of Privatization 257
3.5 Last Value Assignment 258
4 Demand-Driven Symbolic Analysis 261
4.1 Gated Single Assignment 263
4.2 Demand-Driven Backward Substitution 264
4.3 Backward Substitution in the Presence of Gating Functions ... 266
4.4 Examples of Backward Substitution 267
4.5 Bounds of Symbolic Expression 269
4.6 Comparison of Symbolic Expressions 269
4.7 Recurrence and the μ Function 272
4.8 Bounds of Monotonic Variables 273
4.9 Index Array 274
4.10 Conditional Data Flow Analysis 275
4.11 Implementation and Experiments 276
5 Related Work 277

Section III : Communication Optimizations

Chapter 9. Optimal Tiling for Minimizing Communication in Distributed Shared-Memory Multiprocessors
Anant Agarwal, David Kranz, Rajeev Barua, and Venkat Natarajan ... 285

1 Introduction 285
1.1 Contributions and Related Work 286
1.2 Overview of the Paper 288
2 Problem Domain and Assumptions 289
2.1 Program Assumptions 289
2.2 System Model 291
3 Loop Partitions and Data Partitions 292
4 A Framework for Loop and Data Partitioning 295
4.1 Loop Tiles in the Iteration Space 296
4.2 Footprints in the Data Space 298
4.3 Size of a Footprint for a Single Reference 300
4.4 Size of the Cumulative Footprint 304
4.5 Minimizing the Size of the Cumulative Footprint 311
5 General Case of **G** 314
5.1 **G** Is Invertible, but Not Unimodular 314
5.2 Columns of **G** Are Dependent and the Rows Are Independent . 316
5.3 The Rows of **G** Are Dependent 316
6 Other System Environments 318
6.1 Coherence-Related Cache Misses 318
6.2 Effect of Cache Line Size 320
6.3 Data Partitioning in Distributed-Memory Multicomputers 320

7 Combined Loop and Data Partitioning in DSMs 322
7.1 The Cost Model 322
7.2 The Multiple Loops Heuristic Method 325
8 Implementation and Results 328
8.1 Algorithm Simulator Experiments 330
8.2 Experiments on the Alewife Multiprocessor 330
9 Conclusions 334
A A Formulation of Loop Tiles Using Bounding Hyperplanes 337
B Synchronization References 337

Chapter 10. Communication-Free Partitioning of Nested Loops
Kuei-Ping Shih, Chua-Huang Huang, and Jang-Ping Sheu 339

1 Introduction 339
2 Fundamentals of Array References 341
2.1 Iteration Spaces and Data Spaces 342
2.2 Reference Functions 343
2.3 Properties of Reference Functions 343
3 Loop-Level Partitioning 347
3.1 Iteration and Data Spaces Partitioning – Uniformly Generated References 347
3.2 Hyperplane Partitioning of Data Space 353
3.3 Hyperplane Partitioning of Iteration and Data Spaces 359
4 Statement-Level Partitioning 365
4.1 Affine Processor Mapping 366
4.2 Hyperplane Partitioning 372
5 Comparisons and Discussions 377
6 Conclusions 381

Chapter 11. Solving Alignment Using Elementary Linear Algebra
Vladimir Kotlyar, David Bau, Induprakas Kodukula, Keshav Pingali, and Paul Stodghill 385

1 Introduction 385
2 Linear Alignment 388
2.1 Equational Constraints 388
2.2 Reduction to Null Space Computation 390
2.3 Remarks 391
2.4 Reducing the Solution Basis 392
3 Affine Alignment 393
3.1 Encoding Affine Constraints as Linear Constraints 393
4 Replication 396
4.1 Formulation of Replication 397

5 Heuristics ... 398
5.1 Lessons from Some Common Computational Kernels ... 399
5.2 Implications for Alignment Heuristic ... 402
6 Conclusion ... 402
A Reducing the Solution Matrix ... 404
A.1 Unrelated Constraints ... 404
A.2 General Procedure ... 405
B A Comment on Affine Encoding ... 408

Chapter 12. A Compilation Method for Communication–Efficient Partitioning of DOALL Loops
Santosh Pande and Tareq Bali ... 413

1 Introduction ... 413
2 DOALL Partitioning ... 414
2.1 Motivating Example ... 415
2.2 Our Approach ... 419
3 Terms and Definitions ... 421
3.1 Example ... 422
4 Problem ... 423
4.1 Compatibility Subsets ... 423
4.2 Cyclic Directions ... 424
5 Communication Minimization ... 427
5.1 Algorithm : Maximal Compatibility Subsets ... 427
5.2 Algorithm : Maximal Fibonacci Sequence ... 428
5.3 Data Partitioning ... 428
6 Partition Merging ... 429
6.1 Granularity Adjustment ... 431
6.2 Load Balancing ... 431
6.3 Mapping ... 432
7 Example : Texture Smoothing Code ... 432
8 Performance on Cray T3D ... 435
8.1 Conclusions ... 440

Chapter 13. Compiler Optimization of Dynamic Data Distributions for Distributed-Memory Multicomputers
Daniel J. Palermo, Eugene W. Hodges IV, and Prithviraj Banerjee ... 445

1 Introduction ... 445
2 Related Work ... 447
3 Dynamic Distribution Selection ... 449
3.1 Motivation for Dynamic Distributions ... 449
3.2 Overview of the Dynamic Distribution Approach ... 450
3.3 Phase Decomposition ... 451
3.4 Phase and Phase Transition Selection ... 457

4 Data Redistribution Analysis 462
4.1 Reaching Distributions and the Distribution Flow Graph 462
4.2 Computing Reaching Distributions 463
4.3 Representing Distribution Sets 464
5 Interprocedural Redistribution Analysis 465
5.1 Distribution Synthesis 467
5.2 Redistribution Synthesis 468
5.3 Static Distribution Assignment (SDA) 471
6 Results 472
6.1 Synthetic HPF Redistribution Example 473
6.2 2-D Alternating Direction Implicit (ADI2D) Iterative Method . 475
6.3 Shallow Water Weather Prediction Benchmark 478
7 Conclusions 480

Chapter 14. A Framework for Global Communication Analysis and Optimizations
Manish Gupta 485

1 Introduction 485
2 Motivating Example 487
3 Available Section Descriptor 488
3.1 Representation of ASD 490
3.2 Computing Generated Communication 492
4 Data Flow Analysis 494
4.1 Data Flow Variables and Equations 495
4.2 Decomposition of Bidirectional Problem 498
4.3 Overall Data-Flow Procedure 499
5 Communication Optimizations 505
5.1 Elimination of Redundant Communication 505
5.2 Reduction in Volume of Communication 506
5.3 Movement of Communication for Subsumption and for Hiding Latency 507
6 Extensions: Communication Placement 508
7 Operations on Available Section Descriptors 510
7.1 Operations on Bounded Regular Section Descriptors 512
7.2 Operations on Mapping Function Descriptors 514
8 Preliminary Implementation and Results 516
9 Related Work 519
9.1 Global Communication Optimizations 519
9.2 Data Flow Analysis and Data Descriptors 520
10 Conclusions 521

Chapter 15. Tolerating Communication Latency through Dynamic Thread Invocation in a Multithreaded Architecture
Andrew Sohn, Yuetsu Kodama, Jui-Yuan Ku, Mitsuhisa Sato, and Yoshinori Yamaguchi 525

1 Introduction 525
2 Multithreading Principles and Its Realization 527
2.1 The Principle 527
2.2 The EM-X Multithreaded Distributed-Memory Multiprocessor . 530
2.3 Architectural Support for Fine-Grain Multithreading 533
3 Designing Multithreaded Algorithms 535
3.1 Multithreaded Bitonic Sorting 535
3.2 Multithreaded Fast Fourier Transform 538
4 Overlapping Analysis 540
5 Analysis of Switches 544
6 Conclusions 547

Section IV : Code Generation

Chapter 16. Advanced Code Generation for High Performance Fortran
Vikram Adve and John Mellor-Crummey 553

1 Introduction 553
2 Background: The Code Generation Problem for HPF 556
2.1 Communication Analysis and Code Generation for HPF 556
2.2 Previous Approaches to Communication Analysis and Code Generation 558
3 An Integer Set Framework for Data-Parallel Compilation 561
3.1 Primitive Components of the Framework 561
3.2 Implementation of the Framework 562
4 Computation Partitioning 565
4.1 Computation Partitioning Models 565
4.2 Code Generation to Realize Computation Partitions 567
5 Communication Code Generation 573
5.1 Communication Generation with Message Vectorization and Coalescing 577
5.2 Recognizing In-Place Communication 581
5.3 Implementing Loop-Splitting for Reducing Communication Overhead 582
6 Control Flow Simplification 584
6.1 Motivation 584
6.2 Overview of Algorithm 588
6.3 Evaluation and Discussion 589
7 Conclusions 590

Chapter 17. Integer Lattice Based Methods for Local Address Generation for Block-Cyclic Distributions
J. Ramanujam 597

1 Introduction 597
2 Background and Related Work 599
2.1 Related Work on One-Level Mapping 600
2.2 Related Work on Two-Level Mapping 602
3 A Lattice Based Approach for Address Generation 603
3.1 Assumptions 603
3.2 Lattices 604
4 Determination of Basis Vectors 605
4.1 Basis Determination Algorithm 607
4.2 Extremal Basis Vectors 609
4.3 Improvements to the Algorithm for $s < k$ 612
4.4 Complexity 613
5 Address Sequence Generation by Lattice Enumeration 614
6 Optimization of Loop Enumeration: `GO-LEFT` and `GO-RIGHT` 616
6.1 Implementation 620
7 Experimental Results for One-Level Mapping 620
8 Address Sequence Generation for Two-Level Mapping 626
8.1 Problem Statement 626
9 Algorithms for Two-Level Mapping 628
9.1 *Itable:* An Algorithm That Constructs a Table of Offsets 629
9.2 Optimization of the *Itable* Method 631
9.3 Search-Based Algorithms 634
10 Experimental Results for Two-Level Mapping 635
11 Other Problems in Code Generation 638
11.1 Communication Generation 639
11.2 Union and Difference of Regular Sections 640
11.3 Code Generation for Complex Subscripts 640
11.4 Data Structures for Runtime Efficiency 640
11.5 Array Redistribution 641
12 Summary and Conclusions 641

Section V : Task Parallelism, Dynamic Data Structures and Run Time Systems

Chapter 18. A Duplication Based Compile Time Scheduling Method for Task Parallelism
Sekhar Darbha and Dharma P. Agrawal 649

1 Introduction 649
2 STDS Algorithm 652
2.1 Complexity Analysis 663

3 Illustration of the STDS Algorithm 664
4 Performance of the STDS Algorithm 670
4.1 *CRC* Is Satisfied 670
4.2 Application of Algorithm for Random Data 672
4.3 Application of Algorithm to Practical DAGs 674
4.4 Scheduling of Diamond DAGs 675
4.5 Comparison with Other Algorithms 680
5 Conclusions 680

Chapter 19. SPMD Execution in the Presence of Dynamic Data Structures
Rajiv Gupta 683

1 Introduction 683
2 Language Support for Regular Data Structures 684
2.1 Processor Structures 685
2.2 Dynamic Data Structures 685
2.3 Name Generation and Distribution Strategies 688
2.4 Examples 689
3 Compiler Support for Regular Data Structures 693
3.1 Representing Pointers and Data Structures 693
3.2 Translation of Pointer Operations 694
4 Supporting Irregular Data Structures 703
5 Compile-Time Optimizations 705
6 Related Work 706

Chapter 20. Supporting Dynamic Data Structures with Olden
Martin C. Carlisle and Anne Rogers 709

1 Introduction 709
2 Programming Model 711
2.1 Programming Language 711
2.2 Data Layout 711
2.3 Marking Available Parallelism 714
3 Execution Model 715
3.1 Handling Remote References 715
3.2 Introducing Parallelism 718
3.3 A Simple Example 719
4 Selecting Between Mechanisms 722
4.1 Using Local Path Lengths 723
4.2 Update Matrices 724
4.3 The Heuristic 726

5 Experimental Results 731
5.1 Comparison with Other Published Work 733
5.2 Heuristic Results 733
5.3 Summary 735
6 Profiling in Olden 735
6.1 Verifying Local Path Lengths 736
7 Related Work 739
7.1 Gupta's Work 741
7.2 Object-Oriented Systems 741
7.3 Extensions of C with Fork-Join Parallelism 743
7.4 Other Related Work 744
8 Conclusions 745

Chapter 21. Runtime and Compiler Support for Irregular Computations
Raja Das, Yuan-Shin Hwang, Joel Saltz, and Alan Sussman 751

1 Introduction 751
2 Overview of the CHAOS Runtime System 753
3 Compiler Transformations 758
3.1 Transformation Example 759
3.2 Definitions 763
3.3 Transformation Algorithm 765
4 Experiments 769
4.1 Hand Parallelization with CHAOS 769
4.2 Compiler Parallelization Using CHAOS 773
5 Conclusions 775

Author Index 779