## Contents

Series Foreword xiii  
Foreword xv  
Preface xix  

1 Introduction 1  
1.1 Why Parallel Computers Are Here to Stay 1  
1.2 Shared-Memory Parallel Computers 3  
1.2.1 Cache Memory Is Not Shared 4  
1.2.2 Implications of Private Cache Memory 6  
1.3 Programming SMPs and the Origin of OpenMP 6  
1.3.1 What Are the Needs? 7  
1.3.2 A Brief History of Saving Time 7  
1.4 What Is OpenMP? 8  
1.5 Creating an OpenMP Program 9  
1.6 The Bigger Picture 11  
1.7 Parallel Programming Models 13  
1.7.1 Realization of Shared- and Distributed-Memory Models 14  
1.8 Ways to Create Parallel Programs 15  
1.8.1 A Simple Comparison 16  
1.9 A Final Word 21  

2 Overview of OpenMP 23  
2.1 Introduction 23  
2.2 The Idea of OpenMP 23  
2.3 The Feature Set 25  
2.3.1 Creating Teams of Threads 25  
2.3.2 Sharing Work among Threads 26  
2.3.3 The OpenMP Memory Model 28  
2.3.4 Thread Synchronization 29  
2.3.5 Other Features to Note 30  
2.4 OpenMP Programming Styles 31
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>Correctness Considerations</td>
<td>32</td>
</tr>
<tr>
<td>2.6</td>
<td>Performance Considerations</td>
<td>33</td>
</tr>
<tr>
<td>2.7</td>
<td>Wrap-Up</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td><strong>Writing a First OpenMP Program</strong></td>
<td>35</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Matrix Times Vector Operation</td>
<td>37</td>
</tr>
<tr>
<td>3.2.1</td>
<td>C and Fortran Implementations of the Problem</td>
<td>38</td>
</tr>
<tr>
<td>3.2.2</td>
<td>A Sequential Implementation of the Matrix Times Vector Operation</td>
<td>38</td>
</tr>
<tr>
<td>3.3</td>
<td>Using OpenMP to Parallelize the Matrix Times Vector Product</td>
<td>41</td>
</tr>
<tr>
<td>3.4</td>
<td>Keeping Sequential and Parallel Programs as a Single Source Code</td>
<td>47</td>
</tr>
<tr>
<td>3.5</td>
<td>Wrap-Up</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td><strong>OpenMP Language Features</strong></td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>Terminology</td>
<td>52</td>
</tr>
<tr>
<td>4.3</td>
<td>Parallel Construct</td>
<td>53</td>
</tr>
<tr>
<td>4.4</td>
<td>Sharing the Work among Threads in an OpenMP Program</td>
<td>57</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Loop Construct</td>
<td>58</td>
</tr>
<tr>
<td>4.4.2</td>
<td>The Sections Construct</td>
<td>60</td>
</tr>
<tr>
<td>4.4.3</td>
<td>The Single Construct</td>
<td>64</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Workshare Construct</td>
<td>66</td>
</tr>
<tr>
<td>4.4.5</td>
<td>Combined Parallel Work-Sharing Constructs</td>
<td>68</td>
</tr>
<tr>
<td>4.5</td>
<td>Clauses to Control Parallel and Work-Sharing Constructs</td>
<td>70</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Shared Clause</td>
<td>71</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Private Clause</td>
<td>72</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Lastprivate Clause</td>
<td>73</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Firstprivate Clause</td>
<td>75</td>
</tr>
<tr>
<td>4.5.5</td>
<td>Default Clause</td>
<td>77</td>
</tr>
<tr>
<td>4.5.6</td>
<td>Nowait Clause</td>
<td>78</td>
</tr>
</tbody>
</table>
5. How to Get Good Performance by Using OpenMP

5.1 Introduction 125

5.2 Performance Considerations for Sequential Programs 125

5.2.1 Memory Access Patterns and Performance 126

5.2.2 Translation-Lookaside Buffer 128

5.2.3 Loop Optimizations 129

5.2.4 Use of Pointers and Contiguous Memory in C 136

5.2.5 Using Compilers 137

5.3 Measuring OpenMP Performance 138

5.3.1 Understanding the Performance of an OpenMP Program 140
6.4.2 Case Study 3: The Multi-Zone NAS Parallel Benchmarks

6.5 Nested OpenMP Parallelism

6.5.1 Case Study 4: Employing Nested OpenMP for Multi-Zone CFD Benchmarks

6.6 Performance Analysis of OpenMP Programs

6.6.1 Performance Profiling of OpenMP Programs
6.6.2 Interpreting Timing Information
6.6.3 Using Hardware Counters

6.7 Wrap-Up

7 Troubleshooting

7.1 Introduction

7.2 Common Misunderstandings and Frequent Errors

7.2.1 Data Race Conditions
7.2.2 Default Data-Sharing Attributes
7.2.3 Values of Private Variables
7.2.4 Problems with the Master Construct
7.2.5 Assumptions about Work Scheduling
7.2.6 Invalid Nesting of Directives
7.2.7 Subtle Errors in the Use of Directives
7.2.8 Hidden Side Effects, or the Need for Thread Safety

7.3 Deeper Trouble: More Subtle Problems

7.3.1 Memory Consistency Problems
7.3.2 Erroneous Assumptions about Memory Consistency
7.3.3 Incorrect Use of Flush
7.3.4 A Well-Masked Data Race
7.3.5 Deadlock Situations

7.4 Debugging OpenMP Codes

7.4.1 Verification of the Sequential Version
7.4.2 Verification of the Parallel Code
7.4.3 How Can Tools Help?

7.5 Wrap-Up
8 Under the Hood: How OpenMP Really Works 277

8.1 Introduction 277
8.2 The Basics of Compilation 278
  8.2.1 Optimizing the Code 279
  8.2.2 Setting Up Storage for the Program’s Data 280
8.3 OpenMP Translation 282
  8.3.1 Front-End Extensions 283
  8.3.2 Normalization of OpenMP Constructs 284
  8.3.3 Translating Array Statements 286
  8.3.4 Translating Parallel Regions 286
  8.3.5 Implementing Worksharing 291
  8.3.6 Implementing Clauses on Worksharing Constructs 294
  8.3.7 Dealing with Orphan Directives 297
  8.3.8 OpenMP Data Environment 298
  8.3.9 Do Idle Threads Sleep? 300
  8.3.10 Handling Synchronization Constructs 302
8.4 The OpenMP Runtime System 303
8.5 Impact of OpenMP on Compiler Optimizations 304
8.6 Wrap-Up 304

9 The Future of OpenMP 307

9.1 Introduction 307
9.2 The Architectural Challenge 309
9.3 OpenMP for Distributed-Memory Systems 311
9.4 Increasing the Expressivity of OpenMP 312
  9.4.1 Enhancing OpenMP Features 312
  9.4.2 New Features and New Kinds of Applications 314
9.5 How Might OpenMP Evolve? 317
9.6 In Conclusion 318

A Glossary 321

References 331
Index 349