

Using OpenMP

Portable Shared Memory Parallel Programming

Barbara Chapman, Gabriele Jost, Ruud van der Pas

The MIT Press
Cambridge, Massachusetts
London, England

© 2008 Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

This book was set in L^AT_EX by the authors and was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Chapman, Barbara, 1954-

Using OpenMP : portable shared memory parallel programming / Barbara Chapman, Gabriele Jost, Ruud van der Pas.

p. cm. — (Scientific and engineering computation)

Includes bibliographical references and index.

ISBN-13: 978-0-262-53302-7 (paperback : alk. paper)

1. Parallel programming (Computer science) 2. Application program interfaces (Computer software) I. Jost, Gabriele. II. Pas, Ruud van der. III. Title.

QA76.642.C49 2007

005.2'75—dc22

2007026656

Index

Symbols

`#ifdef _OPENMP`, 47
`_OPENMP`
 value of, 47
`atomic` construct
 performance, 147
critical region
 performance, 147
`nowait` clause
 performance, 145
`ordered` construct
 performance, 147
parallel region
 performance, 148
`#pragma omp flush`, 114
`#pragma omp for`, 58
`#pragma omp master`, 94
`#pragma omp ordered`, 86
`#pragma omp parallel`, 53
`#pragma omp section 61`
`#pragma omp sections`, 60
`#pragma omp single`, 64
`#pragma omp threadprivate`, 118
`!$omp do`, 58
`!$omp flush`, 114
`!$omp master`, 94
`!$omp ordered`, 86
`!$omp parallel`, 53
`!$omp section 61`
`!$omp sections`, 60
`!$omp single`, 64
`!$omp threadprivate`, 118
`!$omp workshare`, 66
`barrier` construct, 302
`firstprivate`, 295
`flush`, 302
`lastprivate`, 295
`nowait` clause, 294

A

Active parallel region, 56, 99
Amdahl's law, **33**, 139, 161, 162, 230
ARB, 8
Array padding, 154
Atomic construct, **90**, 91, 92
Atomic operation, 66
Automatic parallelization, **15**
Autoscaling, **314**

B

Barrier, **84**, 85
 incorrect use of, 254
 restrictions on use of, 84
Binding thread set, **53**

C

Cache coherence, **6**, 153, 261
Cache memory, **4**, 29, 125, 126, 128
Cache miss, 126
Cart3D application, **202**
cc-NUMA, **4**, 192, 193
 timings, 144
Chip multithreading (CMT), 309
`chunk_size`, **81**
Clauses, **35**, 70
 copyin, 110, 119
 copyprivate, 110
 default, 77
 firstprivate, 75
 if, 100, 101
 lastprivate, 73
 nowait, 78
 num_threads, 98, 102, **102**, 111
 ordered, 102–104
 private, 72
 processing of, **71**
 reduction, 105–107
 schedule, 79
 shared, 71
Cluster, **11**
Columnwise storage, 128
Combined parallel work-sharing constructs
 as shortcuts, **68**
 benefits of, 69
 clauses for, 68
Computational fluid dynamic applications,
 192
Conditional OpenMP compilation, 47
 sentinel in Fortran, **48**, 49, 50
Constructs, 52
Copyin clause, **110**, 119
Copyprivate clause, **110**
CPU time, **138**
Critical construct, **87**, 88, 89
 name for, 87
Critical section, **87**

D

Data dependence, **27**
 Data dependence analysis, 137
 Data distribution language extensions, 312, 314
 Data parallelism, **192**
 Data race, **244**
 condition, **32**, 73, 122, 244, 245
 debugging, 275
 detection, 275
 Data replication, **205**
 Data reuse pattern, **27**
 Data-sharing attributes, 43, 72
 default rules, 43, 44
 default rules for nested regions, 218
 Deadlock, **268**
 examples, 268
 Debugging, 271
 data race detection, 275
 example session, 273
 sequential version, 271
 tool support, 272
 verification parallel version, 272
 def-sched-var, **97**, 99
 Default clause, **77**, 78
 Default schedule, 97
 directive
 implementation of, 302
 Directives, 9, 25, **35**, 52
 executable, 52
 fixed-source format in Fortran, 36
 free-source format in Fortran, 36
 sentinel in Fortran, 35, 36
 subtle errors, 255
 syntax, 35, 36
 Distributed shared memory (DSM), **11**
 Distributed-memory computers, **11**
 Domain decomposition, 203, **203**, 211
 dyn-var, **97**, 98, 120
 Dynamic number of threads, 97
 Dynamic schedule, 79, 238
 implementation of, 292, 303

E

Efficiency, 139
 Elapsed time, **138**, 230
 Environment variables, **97**
 OMP_DYNAMIC, 98
 OMP_NESTED, 99
 OMP_NUM_THREADS, 97
 OMP_SCHEDULE, 99, 103

EPCC microbenchmarks, 142
 Execution environment, **95**

F

False sharing, **153**, 241, 245
 First Touch, **193**
 Firstprivate clause, **75**, 76
 flowCart application, **201**
 Flush directive, 29, 114, **114**, 115–118
 incorrect use of, 264
 for loops
 extending range of C/C++ loops covered, 317
 Fork-join programming model, **24**
 Fortran array statements, 67, 68

G

gprof, **229**
 Guided schedule, 79

H

Hardware counter, **239**
 cache miss, 240
 instructions, 239
 TLB miss, 240
 Heap storage, **280**, 299
 Hybrid programming, 191, **208**, 221

I

I/O, 56, 89, 103
 Idle threads
 implementation of, 301
 language extension, 318
 If clause, **100**, 101
 Implementation of OpenMP
 on clusters, 311
 Inactive parallel region, 56, 98, 100, 111
 Incremental parallelization, 10
 Initial thread, **24**
 Instruction Level Parallelism, **1**
 ILP, 1
 superscalar architecture, **1**
 Instruction reordering, 279
 Internal control variables, **97**

L

Language extensions, **317**
 automatic scoping, 314
 data distribution features, 312, 314
 data locality, 314
 for loop variables, 317
 idle threads, 318
 loop nest collapsing, 312
 loop schedules, 312, 317, 318
 nested parallelism, 313, 317
 next touch, 314
 task queues, 315
 tasks, 315, 316
 threadstack, 317
 Lastprivate clause, **73**, 74, 75
 Library routines, **97**
 omp_get_dynamic, 98
 omp_get_max_threads, 98
 omp_get_nested, 99
 omp_get_num_procs, 100
 omp_get_num_threads, 99
 omp_get_thread_num, 99, 111
 omp_in_parallel, 100
 omp_set_dynamic, 98
 omp_set_nested, 99
 omp_set_num_threads, 98
 Livelock, **262**
 Load imbalance, 63, 81, 150, 151, 238
 Lock variables, **93**
 declaration of, 93
 Locks, **93**, 94
 caution with, 94
 nestable locks, 93
 simple locks, 93
 Loop
 Fission, **134**, 179
 Fusion, **133**
 Interchange, 129, **132**
 Tiling, **134**
 Blocking, **134**
 Blocking size, 135
 Unroll and jam, **131**, 170, 176, 179
 Unrolling, **129**
 Cleanup loop, **131**
 Unroll factor, **130**
 Loop construct, **58**, 59
 clauses for, 60
 mapping iterations to threads, 60
 restrictions on use of, **58**
 Loop iteration variable
 default attribute, **72**
 Loop nest language extensions, 312

Loop schedules
 implementation of, 292
 language extensions, 312, 317, 318
 Loop-carried dependence, **244**
 Lowering of OpenMP, **282**

M

Master construct, 66, **94**, 95
 special care with, 250
 Master thread, 54, 95
 Memory consistency, **6**, 29, 30, 114, 259
 incorrect assumptions, 262
 Memory fence, **29**
 Memory footprint, 157
 Memory hierarchy, 4, 125–128
 Memory model, 6, **28**, 114, 259
 OpenMP, 260, 261
 thread stack, 29
 Message passing, **13**
 Microbenchmarks, 302
 MPI, **14–18**, 203, 205, 207
 MPI_Init_thread, **210**
 MPI_OPENMP_INTEROP, 197
 mpirun, **222**
 MPPs, **11**
 Multi-zone NAS Parallel Benchmarks, 230
 BT-MZ, 215, 226
 LU-MZ, 215
 SP-MZ, 215
 Multicore, **3**

N

Named critical region, 88
 NanosCompiler, **226**
 NAS Parallel Benchmarks, **211**
 BT, 215
 LU, 215, 239
 Multi-zone, 215
 SP, 215
 nest-var, **97**, 98
 Nestable locks, **93**
 Nested OpenMP, 216, 221
 Nested parallelism, 97, 111, **111**, 112, 113
 language extensions, 313, 317
 Nowait clause, **78**, 79
 position in Fortran, 78
 nthreads-var, 97, **97**, 120
 NUMA, **4**, 193
 numactl, **199**
 Number of threads, **31**, 97

O

omp.h, 47, 97
 OMP_DYNAMIC, 98
 omp_get_dynamic, 98
 omp_get_max_threads, 98
 omp_get_nested, 99
 omp_get_num_procs, 100
 omp_get_num_threads, 99
 omp_get_thread_num, 47, 48, 54, 99, 111
 omp_in_parallel, 100
 omp_lib, 47, 97
 omp_lib.h, 97
 OMP_NESTED, 99
 OMP_NUM_THREADS, 97
 OMP_SCHEDULE, 81, 99, 103
 omp_set_dynamic, 98
 omp_set_nested, 99
 omp_set_num_threads, 98
 OpenMP 2.5, **21**, 47
 OpenMP 3.0, **21**
 OpenMP Architecture Review Board (ARB),
 8
 Operations on atomic construct, 91
 oprofile, **229**
 Ordered clause, 87, **102**, 103, 104
 Ordered construct, **86**, 102–104
 Orphan directives, **30**, 297
 Outlining, 231, **287**
 outlined routines, 232
 Overheads
 load imbalance, 141
 parallel, 139, 141
 sequential, 141
 single thread, 158
 synchronization, 141
 Overlap cells, **203**
 Oversubscribed system, 143
 Owner computes rule, **203**

P

Parallel computer architectures, **11**
 Parallel construct, **53**, 54–57
 active, 56
 clauses for, **55**
 data sharing attributes with, 59
 implementation of, 286
 implicit barrier, 54
 inactive, 56
 number of executing threads, 56
 restrictions on use of, 56
 Parallel efficiency, 235

Parallel loops, **26**, **27**, 58
 iteration variable, 72
 permissible forms in C and C++, 59
 schedule, 79, 81
 Parallel overhead., **139**
 Parallel program design, 10
 Parallel program overheads, **34**, **140**
 Parallel region, 24, 25
 active, 99
 inactive, 98, 100, 111
 number of threads in, 56
 Parallel scalability, **34**
 Parallel sections, **60**
 implementation of, 284
 Parallel speedup, **33**
 Paraver Performance Analysis System, **235**
 PCF, **7**, **8**
 Performance
 array reduction, 182
 false sharing, 154, 178, 181
 private data, 155
 private versus shared data, 156
 Performance Analyzer, 229
 Performance profile, 229
 Pipelined processing, 151
 Pointer aliasing problem, **136**
 POP Ocean Circulation Model, **215**
 Preserving sequential execution, **47**
 Private clause, **72**, 73
 loop iteration variable, 72
 Private data, **28**, 72
 special care with, 249
 undefined on entry to and exit from con-
 struct, 73
 Private variable
 broadcasting value, 110
 Private variables
 reducing number of, 124
 Process, **23**
 Pthreads, **16**, 18, 20

R

Race condition, **32**
 Real time, **138**
 Reduction clause, 105, **105**, 106
 array reductions in Fortran, 107, 109
 rules on use of, 109
 supported operators, 106–108
 Reduction operation, 88, **105**, 106
 explicitly programmed, 89
 Reduction variable, **106**
 Region of code, **53**
 Remote memory access, 240

Replicated work, 240
 Restrict keyword, **38**
 restrict keyword, 137
 Rowwise storage, 127
 run-sched-var, **97**
 Run-time schedule, 81, 97
 Runtime library, 243, 277, **303**, 305

S

Sampling, **229**
 Schedule clause, **79**, 81–83
 Schedule kinds, **79**
 default, 97
 dynamic, 79
 guided, 79
 runtime, 81, 97
 static, 79
 Sections construct, **60**, 61–63, **63**, 64
 clauses for, 64
 Sentinel, **36**
 Sequential consistency, **259**
 Sequential performance, 125
 Shared clause, **71**, 72
 Shared data, 71
 special care with, 246
 Shared-memory model, **13**
 Simple locks, **93**
 Single construct, **64**, 65, 66
 barrier at end of, 64
 clauses for, 66
 implementation of, 285
 SMP, **3**, 4–8, 11
 Software Distributed Shared Memory, **311**
 Software pipelining, **1**
 Speedup, 139
 superlinear, 141, 160, 161, 166, 177
 SPMD, 32, **192**, 200
 Stack, 29, 200
 Static schedule, 79
 implementation of, 292
 Structured block, 25, **52**
 rules in C/C++, 53
 rules in Fortran, 53
 Superlinear speed-up, **328**
 Synchronization, **29**, **83**, 237
 Synchronization points, **30**, 114, 115

T

Task parallelism, **192**
 Taskset, **199**
 Team of threads, 25
 Thread, 3, 13, **23**
 Thread creation, 286
 Thread migration, 194
 Thread number, **31**
 Thread synchronization, 302
 Thread-safe, **255**
 Class objects and methods in C++, 258
 Fortran SAVE, 257
 Library functions, 258
 Threadid, 54
 Threadprivate, 110
 Threadprivate data, 299
 Threadprivate directive, **118**, 119–123
 Threadstack, **299**, 317
 TLB, 4, **128**
 Translation-lookaside buffer, **128**

U

UMA, **3**
 Unit stride, 127
 Useful parallel time, **233**

W

Wall-clock time, **138**, 230
 Work-sharing, 26
 Work-sharing constructs, 26, 57, **57**, 58
 implementation of, 291
 incorrect assumptions, 252
 incorrect nesting, 253
 Workshare construct, **66**, 67, 68
 implementation of, 286
 Workshare duration, **237**

X

X3H5 committee, **7**, **8**