Index

A
abstract data type (ADT), 123, 174, 184, 307
defining, 174–175
definition, 307
implementation, 314
abstraction. See also Distributed Array pattern; SPMD pattern;
Supporting Structures design space
clarity of, 123–124, 128, 183, 200
definition, 307
accumulation, shared data, 47
address space, 157, 220
definition, 307
ADT. See abstract data type
affinity of shared memory to processors, 252
Alexander, Christopher, 4, 5
algorithm, 57
parallel overhead, 143
performance, 32
Algorithm Structure design space, 5–6, 24–27
concurrency, organizing principle, 60–61
decision tree, 60–62
efficiency, 58
patterns, 110
portability, 58
scalability, 58
selection, 59–62
simplicity, 58
target platform, 59
algorithm-level pipelining, 103
Amdahl’s law, 19–22
definition, 307
AND parallelism, definition 307–308
anonymous inner classes, 294
API (application programming interface), 12, 67
definition, 308
usage, 199
architectures, parallel
computers, 8–12
array-based computation, 35
array. See also block-based array
decomposition; Distributed Array pattern; matrix
distributions, standard, 200–205
ASCI Q, 127
assembly line analogy, Pipeline pattern, 103–104
associative memory, 72
asynchronous
computation, 205, 312
communication in MPI, 284
definition, 17
events, 62, 117
interaction, 50, 55, 120
message passing, 17, 117, 146, 284
atomic, 221, 297
definition, 308
OpenMP construct, 231, 257, 264–265
autoboxing, 308
B
bag of tasks, in master/worker
algorithms, 122
implementation, 144
management, 144–146
bandwidth, 21–22. See also bisection bandwidth
definition, 308
bandwidth-limited algorithms, 308
Barnes-Hut algorithm, 78
barrier, 6, 226–229. See also Java; MPI;
OpenMP
definition, 308
impact, 229
OpenMP construct, 228–229
benchmarks, 21
Beowulf cluster, 11
definition, 308
binary hypercube, 312
binary operator, 265
binary tree, 62
bisection bandwidth,
definition, 308
bitwise equivalence, 125
blackboard, 111
block-based array decomposition, 50.
  See also Distributed Array
  pattern; matrix;
  one-dimensional block
  distribution; square chunk
  decomposition; two-
  dimensional block distribution
column block distribution, 200
cyclic distribution, 85, 202, 310
row block distribution, 200
block-based matrix multiplication, 50
block-on-empty queue, 184–188
bottlenecks, performance, 153
branch-and-bound computations, 64
broadcast
  definition, 309
  mechanism, 245
Bruno, Giordano, 254
buffered communication mode, in
  MPI, 286
buffers, 280
bytecode, 291. See Java

C
C# programming language, 15
C programming language, 13, 246,
  254, 320
C++ programming language, 13, 246,
  254, 320
cache, 10, 38, 87, 157
  coherency protocols, 267
  definition, 309
cache lines, 157
  definition, 309
  invalidate-and-movement
  operations, 158
cache-coherent NUMA, 10
  definition, 309
Calypso, 151
car-wash example, discrete-event
  simulation, 115
causality, 118
ccNUMA. See cache-coherent NUMA
Celera Corp., 2
central processing units (CPUs)
  hardware, 103
  instruction pipeline, 103
  time, 2

Chain of Responsibility pattern
  (COR), 113
ChiliPLoP, 4
Cholesky decomposition, 73
chunks. See also block-based array
decomposition
decomposition, 79–82
mapping onto UEs, 200
redistribution, 85
Cilk, 69
clarity of abstraction. See abstraction
client-server computing, 214
clusters, 8, 15
Co-Array Fortran, 252
course-grained data parallelism, 79
code transformations, 66–67
collective communication, 245–251
  definition, 309
  operations, 13, 245
collective operations. See MPI
column block distribution
decomposition, 206
Distributed Array pattern example,
  207–211
combinatorial optimization, 102
communication, 21–22, 237–252. See
  also collective communication;
  global communication;
  nonblocking communication;
  overlapping communication
  and computation
APIs, 84
costs, 77
efficiency, 92
mode in MPI, 91 (see also buffered
  communication mode;
  immediate communication
  mode; ready communication
  mode; standard
  communication mode;
  synchronous communication
  mode)
one-sided, 15
overhead, 53, 64
communication context, in MPI, 226–228
communicator, in MPI, 226
compiler optimizations, 178, 222
computational chemistry, 73
computational geometry, 78
computational linear algebra, 73, 79
computer animation, 1
concurrency. See also fine-grained concurrency; Finding
Concurrency design space
exposure, 99
finding, 5
in operating systems, 7–8
understanding, 61
concurrency-control protocol, 175. See also noninterfering operations;
one-at-a-time execution;
readers/writers
consideration, 183
implementation, 175–178, 181
nested locks, usage, 188–190
concurrent computation, 67
concurrent execution, 39
definition, 309
concurrent programming, 15–16. See also parallel programming
content-addressable memory, 72,
117, 252
c conditioning variable, 299, 301, 310
cyclic distribution, 85, 138. See also
loop iterations; block-based array decomposition
definition, 310

D
data
access protection, 47
chunk, 45
environment clauses in OpenMP, 262–265

exchange, 82–85, 303
mapping onto UEs, 200
replication, 37, 72, 174
data decomposition, 25. See also
block-based array decomposition; Divide and
Conquer pattern; Recursive
Data pattern
dimension, 26
efficiency, 35–36
flexibility, 35–36
organization, 61
usage, 45

Data Decomposition pattern, 25, 27, 29,
34–39
c context, 34
e examples, 36–39
in other patterns, 29, 31, 32, 34,
42, 45, 50, 63
forces, 35
problem, 34
solution, 35
data dependency, 45, 135, 249
data distribution. See also Distributed
Array pattern
block, 200
block-cyclic, 200
choosing, 206
cyclic, 200
data flow
irregular, 61
organization, 61
representation in the Pipeline
pattern, 107–108
data parallel. See also coarse-grained
data parallelism
algorithms, 79, 101
definition, 310
data sharing, 25. See also fine-grained
data sharing; tasks
analysis, 34
overhead, 46

Data Sharing pattern, 25–26,
44–49
c context, 44–45
e examples, 47–49
in other patterns, 34, 39, 134
forces, 45–46
problem, 44
solution, 46–47
data structures, 28, 79–80. See also recursive data structures
construction, 43
management, 175
pattern representation, 123
data-driven decomposition, 31
dataflow languages, 312
deadlock, 18, 279
avoidance, 118–119, 177
definition, 310
prevention, 181
production, 210
declarative languages, 214–215
decomposition. See also data
decomposition; data-driven decomposition; functional
decomposition; row-based block decomposition;
task-based decomposition
patterns, 25
examples, background, 26–29
usage, 26
flexibility, 52
dependencies. See also loop-carried
dependencies; removable
dependencies; separable
dependencies; Task Parallelism pattern; temporal
dependencies
analysis, 25, 39, 49, 57
simplification, 40
categorization, 66–68
data sharing
removable, 66–67
separable, 67
handling, 46
management, 30
ordering constraints, 40
regularity, 54–55
temporal, 40
types, 66–68
dependency analysis patterns,
25–26
Design Evaluation pattern, 25–26,
49–55
context, 49
forces, 49–50
in other patterns, 37
problem, 49
solution, 50–55
design patterns, 4–5
definition, 310
design quality, 50, 52–54
efficiency, 53–54
flexibility, 52–53
simplicity, 54
design spaces, 24–29. See also
Algorithm Structure design
space; Finding Concurrency
design space; Implementation
Mechanisms design space;
Supporting Structures design
space
overview, 25–26
DFT. See discrete Fourier transform
DGEOM. See distance geometry
program
differential equations, 28, 60, 80, 198
digital circuit simulation, 119
direct task/UE mapping, 168–169
usage. See mergesort
directive formats in OpenMP, 257–259
discrete Fourier transform (DFT), 78.
See also Fourier transforms;
inverse DFT
discrete-event simulation, 118
discretization, 80
distance geometry program
(DGEOM), 73
distributed array. See also block-based
array decomposition
computation alignment, 207
distribution selection, 205
indices, mapping, 205–206
Distributed Array pattern, 122–123,
198–211
context, 198–199
examples, 207–211
forces, 199–200
in other patterns, 36, 38, 85, 97,
128, 142–143, 182, 251
problem, 198
related patterns, 211
solution, 200–207
distributed computing, definition, 310
distributed-memory computers,
11–12, 51
architecture, 11, 51
environment, 51
MIMD
architecture, 317
computers, 129, 211
models, 15
systems, 11, 47, 74, 237
distributed queue, 144
distributed shared memory (DSM). See also virtual distributed shared memory systems
definition, 310–311
divide-and-conquer algorithms, 74. See also sequential
divide-and-conquer algorithms
divide-and-conquer matrix multiplication. See parallel
divide-and-conquer matrix multiplication
Divide and Conquer pattern, 58, 61, 73–79
class, 73
data decomposition, 82–83
forces, 74
in other patterns, 97, 125, 126, 127, 146, 167, 168, 173
problem, 73
related patterns, 78–79
solution, 75–77
divide-and-conquer strategy. See divide-and-conquer algorithms
DNA sequencing, 1–2
domain decomposition, 79
double-ended task queue, 146
DPAT simulation, 119
DSM. See distributed shared memory
dual-processor computers, 1
dynamic schedule, 69, 271
dynamic load balancing, 161
E
eager evaluation, definition, 311
ear decomposition, 102
Earth Simulator Center, 127
efficiency
definition, 124, 311
portability, conflict, 58
eigenvalues/eigenvectors, 78
Einstein, Albert, 54
embarrassingly parallel, 60
definition, 311
problems, 70
Ensemble system for discrete-event simulation, 118
environmental affinity, 125
equal-time memory access, 318–319
error condition handling, 108
Ethernet LAN clusters, 134
Ethernet network, 11
Euler tours, 102
European Workshop on OpenMP (EWOMP), 166
EuroPLoP, 5
Event-Based Coordination pattern, 58, 61–62, 114–120
class, 115–116
elements, 119
forces, 116
in other patterns, 114
problem, 114–115
related patterns, 120
scheduling, 119
solution, 116–119
tasks, defining, 116
events
communication, efficiency, 119
flow, representation, 117
ordering, 117–118
EWOMP. See European Workshop on OpenMP
elements
computing Fibonacci numbers, 194–196
Fourier transform computation, 109–110
heat diffusion (see mesh computation)
image construction, 70–71
linear algebra, 27, 34, 207
loop-based programs in Java, 308–309
Mandelbrot set generation (see Mandelbrot set)
matrix diagonalization, 78
matrix transpose, 207–210
medical imaging, 26, 31, 36, 62–63
mergesort, 77, 169–171
mesh computation, 80, 83, 85–92, 164–166
examples (cont.)
    NUMA computers, 164
    OpenMP, 164
    molecular dynamics, 27–29, 32–34,
        38–39, 41–42, 44, 47–49,
        63–64, 71–72, 133–140,
        160–161
    numerical integration, 129–133,
        159–160
    MPI, 130
    OpenMP, 160
    partial sums of linked lists, 101
    pipeline framework, 110
    sorting pipeline, 113
    timing a function using a barrier,
        227–230
    exchange, data, 84
    explicitly parallel language
        definition, 311
    exploitable concurrency, 3, 63,
        80, 245
    exposing concurrency, 24
    extraterrestrial intelligence, search for.
        See Search for Extraterrestrial
            Intelligence

F
Facade pattern, 116
Factory pattern
    definition, 311
    interaction, 295
false sharing, definition, 311
Fast Fourier Transform (FFT), 66. See
    also Fourier transforms
fast multipole algorithm, 78–79
fault-tolerant computing, 147
fences. See Java; MPI; OpenMP
FFT. See Fast Fourier Transform
Fibonacci number computation, Shared
    Queue pattern example,
        194–196
file systems, parallel, 108
Finding Concurrency design space,
    5–6, 24
fine-grained concurrency, 8, 101,
    106, 172
fine-grained data sharing, 51
fine-grained parallelism. See
    fine-grained concurrency
finite element methods, 141
finite differencing scheme, 97
first-order predicate calculus, 214
firstprivate clause, in
    OpenMP, 164
first touch page placement, 164
fixed-form Fortran statements, in
    OpenMP, 258
fixed-time speedup, 21
FJTask
    framework, 69, 77, 171
    objects, 171
    package, 169, 171, 190
    subclass, 171
FLAME project, 78
floating-point arithmetic, 3, 32
    associativity, 248
    operations, 38
flush construct, in OpenMP, 233
Flynn’s taxonomy
    MIMD, 9
    MISD, 9
    SIMD, 8
    SISD, 8
for construct, in OpenMP, 76
fork-join programming model, in
    OpenMP, 172
fork/join
    approach, 76
    definition, 167–173, 311
    programs, 77
Fork/Join pattern, 122–123, 125–126,
    167–173
    context, 167–168
    examples, 169–173
    forces, 168
    in other patterns, 76, 78, 143, 147,
        152, 197
    problem, 167
    related patterns, 173
    solution, 168–169
    thread-pool-based
        implementation, 197
Fortran. See also High Performance
Fortran
    usage with MPI, 288–289
    usage with OpenMP, 253–271
Fourier transforms. 66, 75, 78, 97
Pipeline pattern example, 109–110
framework, definition, 311
functional decomposition, 30
functional programming languages, 215
future variable, definition, 312
Fx (language), 14, 111

G
GAFORT program, 179
GAMESS, 73
Gamma, Erich. See Gang of Four.
Gang of Four (GoF), 4
GA. See Global Arrays
Gaussian Quadrature. See recursive
Gaussian Quadrature
generic type, 292, 305
generics, definition, 312
genetic algorithm. See nonlinear
optimization
Geometric Decomposition pattern, 58,
   61, 79–97
cost, 79–81
examples, 85–97
forces, 81
in other patterns, 38, 64, 102, 111,
   125–127, 142, 153, 164–165,
   167, 173, 199, 211, 215
problem, 79
related patterns, 97
solution, 82–85
Georgia Tech Time Warp (GTW), 119
ghost cells, 88
Global Arrays (GA), 15, 252
global communication, 144
global data array, 88
global indices, 205
   mapping, 201
global optimization, 199
GoF. See Gang of Four
granularity, 36, 82
granularity knobs, 36
graphical user interface (GUI), 8, 291
graphics, 103–104
grids, 12
   computations (see SETI@home)
definition, 312
technology, 214
Gröbner basis program, 181
Group Tasks pattern, 25, 39–42
cost, 39–40
examples, 41–42
in other patterns, 32, 42, 44, 45,
   47, 59, 55

H
Haskell, 215
heat diffusion. See mesh computation
heavyweight object, 217
Helm, Richard. See Gang of Four.
heterogeneous systems, definition, 312
hierarchical task group, 55
High Performance Fortran (HPF),
   100–101, 212
language, 211
usage, 111–112
high-level synchronization constructs,
   223
high-level programming language, 216
high performance computing (HPC),
   15, 16
Hillis, Daniel, 101, 102
Hillside Group, 4–5
Hoare, C. A. R., 314
homogeneous system, 68
Hood, 69
HPF. See High Performance Fortran
HTTP request
   filtering, 113
   handling, 114
Hubble Space Telescope, 110–111
hybrid computer architectures, 13
hybrid MIMD machine, 12
hypercube, definition, 312
Hyper Threading, 318

I
image processing applications, 65
immediate communication mode, in
   MPI, 91
Implementation Mechanisms design
   space, 154
implicit barrier, in OpenMP, 229, 262
implicitly parallel language
definition, 312
incremental parallelism (refactoring),
   153, 253–254
definition, 312
independent tasks, 29
indices, mapping. See Distributed Array pattern
indirect task/UE mapping, 169
Inmos Ltd., 319
input/output (I/O). See also parallel I/O; nonblocking I/O
   delays, 8
   facilities, 244
   library, 164
   operations, 43
   thread-safe, 255, 266
instruction pipeline, 103
Intel Corporation, 18
invalidate-and-movement operations.
   See cache line
inverse DFT, 109. See also Fourier transforms
I/O. See input/output
irregular decomposition, 55
irregular interactions, 50, 55, 61–62, 114–115
iterations. See loop iterations
iterative constructs, 152

J
J2EE. See Java 2 Enterprise Edition; 214
JáJá, Joseph, 102
Jacobi iteration, 171
Java
   anonymous inner classes, 294
   atomic array, 225
   autoboxing, 292, 308
   barrier, 229
   blocking I/O, 244
   buffers, 244
   bytecode, 312
   channel, 244
   classes (see Java classes)
   comparison with OpenMP, 303
   concurrent programming, 292
   usage, 193, 308, 315
daemon thread, 293
factory, 294
fences, 225–226
final variables, 293
floating-point arithmetic model, 3, 16
generic types, 292
interfaces (see Java interfaces)
interrupts, 304
language definition, 241
memory synchronization, 178
message passing, 241–246
MPI-like bindings, 244
mutual exclusion, 233–236
performance, 241–246
pipeline framework, Pipeline pattern example, 110
portability, 58
process creation/ destruction, 220–221
run method, 148
scope rules, 218, 293
synchronized blocks, 297–298
TCP/IP support, 244
thread creation/ destruction, 218
visibility, 225
volatile, 178
wait and notify, 185
wait set, 299
with distributed-memory systems, 15, 221

Java classes
   AtomicLong, 297
   Buffer, 244
   ConcurrentHashMap, 304
   CopyOnWriteArrayList, 304
   CopyOnWriteArraySet, 304
   CountDownLatch, 110
   CyclicBarrier, 229
   Exchanger, 303
   Executors, 148
   Future, 295, 312
   java.lang.Process, 221
   java.lang.Runtime, 221
   LinkedBlockingQueue, 110
   Object, 298
   ReentrantLock, 301
   Thread, 319
   ThreadPoolExecutor, 148

Java interfaces
   BlockingQueue, 110
   Callable, 295
   Collection, 304
   Condition, 303
   Executor, 148, 294–296
   Executors, 294
   Runnable, 148–149, 293–296
Java Native Interface (JNI), 244
Java Virtual Machine (JVM), 244
  definition, 312
  implementation, 221
  specification, 297
java.io (package), 242
java.lang (package), 292
JavaMPI, 244
java.net (package), 243
java.nio (package), 244
java.rmi (package), 242
JavaSpaces, 117, 252, 320
java.util (package), 292
java.util.concurrent (package), 148, 176, 183–185, 219, 229, 292, 294, 303
  usage
java.util.concurrent.atomic (package), 297
java.util.concurrent.lock (package), 233, 293
Java 2 Enterprise Edition, 113
JNI. See Java Native Interface
Johnson, Ralph. See Gang of Four.
join. See fork/join
JVM. See Java Virtual Machine

K
KoalaPLoP, 4

L
LAM/MPI, 213, 273
LAPACK, 172
large-grained tasks, 42
Last in First Out (LIFO) buffer, 190
lastprivate clause, in OpenMP, 264
latency, 21, 109
  cost, 53
  definition, 313
  hiding, 22
latency-bound algorithms, 313
lazy evaluation, definition, 313
LIFO. See Last In First Out
local data, 45. See also task-local data
  identification, 49
local indices, mapping, 202–205
local variable, 263
locality, definition, 313
locks, 47, 301–303
  acquisition, 181
  functions, 266
logic programming, 214–215
logic programming languages, 214–215
loop iterations, 128, 138, 259
  cyclic distribution, 133
  dependency, 67
  independence, 259–260
  interaction, 163
  splitting, 129 (see also loops)
Loop Parallelism pattern, 122–123, 125–126, 152–167
  context, 152
  examples, 159–167
  forces, 153
  in other patterns, 69, 71, 72, 85, 87, 142, 143, 146, 149, 151, 167, 169, 172, 173, 180, 259, 263
  performance considerations, 157–158
  problem, 152
  related patterns, 167
  solution, 153–158
loop-based parallelism, 125
  algorithms, 157
  performance, 157

linear speedup, 19
linked lists, Recursive Data pattern
  example, 102
Linux operating system, 11
LISP, 215
list-ranking, Recursive Data pattern
  example, 102
load balance, 17, 50, 199
  definition, 313
  improvement, 85
  support, 145
load balancing, 17, 119
  definition, 313
  difficulty, 143
  facilitation, 82
  problem, 30
  statistical, 71
local, 45. See also task-local data
  identification, 49
local indices, mapping, 202–205
local variable, 263
locality, definition, 313
locks, 47, 301–303
  acquisition, 181
  functions, 266
logic programming, 214–215
logic programming languages, 214–215
loop iterations, 128, 138, 259
  cyclic distribution, 133
  dependency, 67
  independence, 259–260
  interaction, 163
  splitting, 129 (see also loops)
Loop Parallelism pattern, 122–123, 125–126, 152–167
  context, 152
  examples, 159–167
  forces, 153
  in other patterns, 69, 71, 72, 85, 87, 142, 143, 146, 149, 151, 167, 169, 172, 173, 180, 259, 263
  performance considerations, 157–158
  problem, 152
  related patterns, 167
  solution, 153–158
loop-based parallelism, 125
  algorithms, 157
  performance, 157
loop-carried dependencies, 66, 152
removal, 152
loop-driven problems, 153
loop-level pipelining, 103
loop-level worksharing constructs, in
OpenMP, 123
loops. See also parallelized loop;
time-critical loops
coalescing (see nested loops)
merging, 155
parallel logic, 167
parallelism, 122, 154
parallelization, prevention, 67
range, 130
schedule, optimization, 154
sequence, 154
splitting, 129, 259
strategy, 131
structure, 94
loop-splitting algorithms, 31
Los Alamos National Lab, 127
low-latency networks, 66
low-level synchronization protocols, 267
LU matrix decomposition, 172
M
maintainability, 124
Mandelbrot set, 70–71. See also parallel
Mandelbrot set generation
generation, 147
Loop Parallelism pattern example,
159–167
Master/Worker pattern example,
147–151
SPMD pattern example, 129–142
mapping data to UEs, 200
mapping tasks to UEs, 76–77
MasPar, 8
massively parallel processor (MPP)
computers, 8, 11
definition, 313–314
vendors, 314
master thread in OpenMP, 168
master/worker algorithm, 144
Master/Worker pattern, 122–123,
125–126, 143–152
completion detection, 145–146
context, 143
examples, 147–151
forces, 144
in other patterns, 70, 71, 72, 76,
167, 173, 183, 188, 219, 319
problem, 143
related patterns, 151–152
solution, 144–147
variations, 146
matrix
blocks, 81
indices, 95
order, 201
matrix diagonalization, 78
Divide and Conquer pattern
example, 78–79
matrix transposition
Distributed Array pattern example,
207–211
matrix multiplication, 16. See also
parallel divide-and-conquer
matrix multiplication; parallel
matrix multiplication
algorithm (see also block-based
matrix multiplication
algorithm)
complexity, 27
Data Decomposition pattern
example, 36–37
Geometric Decomposition pattern
example, 85–97
Group Tasks pattern example,
41–42
problem, 39
Task Decomposition pattern
example, 31–34
medical imaging, 26–27
Algorithm Structure design space
example, 62–64
Finding Concurrency design space
example, 36–37
Task Decomposition pattern
example, 31–34
memory
allocation, 282
bandwidth, 10
bottleneck, 10
busses, speed, 199
fence, 222
hierarchy, 239
usage, 198
management, 200
model, description, 293
Index 345

subsystem, 51
synchonization, 178, 297
    fences, interaction, 221–226
    guarantee, 233
    utilization, 153
mergesort
    Divide and Conquer pattern
    example, 78
Fork/Join pattern example, 76–78, 169–172
mesh computation
    ghost boundary, 83
    Loop Parallelism pattern example, 164–166
    Geometric Decomposition pattern
    example, 80, 85–92
    in OpenMP, 87–88
    in MPI, 88–92
    NUMA computers, 35, 164, 273
message buffer, in MPI, 117
message passing, 6, 238–245. See also
    asynchronous message
    passing; Java; MPI; Message Passing Interface; OpenMP;
    point-to-point message
    passing
design, 52
environment, 51, 107, 117, 175, 249
functions, 275
Java, 288
OpenMP, 240–241
usage, 86
Message Passing Interface (MPI), 13, 84
    API, 220
    barriers, 226–229
    collective operations, 279–284
    concepts, 273–275
    defintion, 314
    fences, memory, 226
    Fortran language binding, 288–289
    Forum (see Message Passing Interface Forum)
    implementation, 213
    LAM/MPI, 213
    MPICH, 213
    initialization, 275–277
    introduction, 273
    Java bindings, 273
    message passing, 238–241
    mutual exclusion, 236–237
nonblocking communication, 84,
    90, 284, 286
persistent communication, 286
process creation/destruction, 220–221
thread creation/destruction, 218–220
timing functions, 281
Version 2.0, 15
Message Passing Interface (MPI) Forum, 15
middleware, 12, 214
MIMD. See Multiple Instruction Multiple Data
MISD. See Multiple Instruction Single Data
molecular dynamics, 27–29
    Algorithm Structure example, 62–63
    Data Decomposition pattern
    example, 36–39
    Data Sharing pattern example, 47–49
    Group Tasks pattern example, 41–42
    Loop Parallelism pattern example, 159–167
    Order Tasks pattern example, 44
    SPMD pattern example, 129–141
    Task Decomposition pattern
    example, 31–34
    Task Parallelism pattern example, 70–73
    WESDYN, 34
monitor, definition, 314
monotonic counter, 144
monotonic indices, 152
Monsters, Inc. (2001), 1
Monte Carlo
    model, 27
    simulation, 50
MPI. See Message Passing Interface
    MPI_Allreduce, 246
    MPI_ANY_TAG, 236
    MPI_BARRIER, 279
    MPI_Bcast, 280
    MPI_Bsend. See buffered communication mode
    MPICH, 213, 273
    MPI_COMM, 226
MPI_Comm_rank, 276, 290
MPI_Comm_size, 276, 290
MPI_COMM_WORLD, 130
MPI_Init, 275
MPI_Irecv, 91
MPI_Isend, 91
mpiJava, 244
MPI_MAX, usage in reductions, 246, 284
MPI_MIN, usage in reductions, 246
MPI_Recv, 290
MPI_Recv_init, 286
MPI_Reduce, 245, 246
MPI_Rsend, 287
MPI_Send, 286
MPI_Send_Init, 286
MPI_Rsend. See ready communication mode
MPI_Send. See standard communication mode
MPI_Start, 286
MPI_Status, 278
MPI_SUM, usage in reduction, 246
MPI_Test, 284
MPI_Wait, 286
MPI_Wtime, 229
mpirun, 275
MPMD. See Multiple Program Multiple Data
MPP. See massively parallel processor
MTA, 22, 51
multicomputer, 312
definition, 314
Multiple Instruction Multiple Data (MIMD), 9, 314, 318. See also hybrid MIMD machine;
distributed memory;
shared memory
definition, 314
Multiple Instruction Single Data (MISD), 9
Multiple Program Multiple Data (MPMD), 212–213
program structure, 126
multiple-read/single-write data, 47
multipole algorithm/computation. See fast multipole algorithm
multiprocessor workstations, 8
multithreaded APIs, 311
multithreaded server-side applications, 291
multithreading, simultaneous, 162, 318
mutex. See mutual exclusion
mutual exclusion (mutex), 175, 229–237. See also Java; MPI; OpenMP
constructs, 230
definition, 314
implementation, 287
usage, 230
N
NASA, 73
native multithreaded APIs, usage, 199
\(N\)-body problem, 28
nearest-neighbor algorithm, 78
nested locks, 177, 181
usage (see concurrency-control protocol)
nested loops, coalescing, 154
nested synchronized blocks, 190
network, 11
bandwidth, 37, 51
infrastructure, 11
networked file systems, usage, 108
newsroom analogy for the Event-Based Coordination pattern, 115
node, 12
definition, 314
loads, 138
number, 65
nonblocking communication, 90, 284
nonblocking I/O, 244
nonblocking queue, 184
nonblocking shared queue, 187
noninterfering operations,
  concurrency-control protocol, 187–188
  noninterfering operations,
  concurrency-control protocol, 187–188
nonlinear optimization using genetic algorithms, Shared Data
pattern example, 179–181
Nonuniform Memory Access (NUMA) computer
definition, 314
cache-coherent NUMA (ccNUMA) machines, 309
page-placement algorithm, 164
platform, 199
times, 13
notify, in Java
Object class method, 300
replacement, 301
notifyAll, in Java
   addition, 186
   Object class method, 300
   replacement, 301
nowait clause, in OpenMP, 261–262
null events, 118
NUMA. See Nonuniform Memory Access
numerical integration
   Loop Parallelism pattern example, 152–167
   SPMD pattern example, 129–133

O
   object-oriented design, 2
   object-oriented frameworks, 107
   object-oriented programming (OOP), 4
      techniques, 107
      usage, 107
   off-the-shelf network, 11
   OMP. See OpenMP
   omp_get_num_threads, 266
   omp_get_thread_num, 266
   omp_lock_t, 181, 232
   OMP_NUM_THREADS, OpenMP
      environment variable, 257
one-at-a-time execution,
   concurrency-control protocol, 175
one-deep divide and conquer, 77
one-dimensional (1D) block
distribution, 200
one-dimensional (1D) block-cyclic
distribution, 200
one-dimensional (1D) differential
equation, 80
one-sided communication, 226
OOP. See object-oriented programming
opaque type, definition, 314
OpenMP (OMP), 13
   API, 223
      barrier construct, 228
      clusters, 15
      comparison with Java, 303
      constructs, 257
      core concepts, 254–257
      critical construct, 268
      data environment clauses, 262–265
      directive formats, in Fortran, 257–259
   definition, 314–315
   DO construct, in Fortran, 261
   fences, 222–225
   firstprivate clause, 264
   flush construct, 223
   for construct, in C and C++, 261
   implementations, 76
   implicit barrier, 229, 261
   lastprivate clause, 264
   lock, 269
   message passing, 239–245
   MPI emulation, 239
   mutual exclusion, 267
   NUMA, 239
   pairwise synchronization, 181
   parallel construct, 255
   parallel for construct, 76
   private clause, 87
   pragma format, C and C++, 258
   process creation/destruction, 221
   reduction clause, 246
   runtime library, 265–266
   schedule clause, 270–272
   sections construct, 262
   single construct, 262
   specifications, 253
   structured block, 255
   synchronization, 266–270
   syntax, 265
   task queue, 319
   thread creation/destruction, 218
   worksharing constructs, 259–262
operating systems
   concurrency (see parallel programs vs operating system concurrency)
   overhead, 53
   optimistic event ordering, 118
   optimizations, 77
   OPUS system, 110
   OR parallelism, definition, 315
Order Tasks pattern, 25–26, 42–44
   context, 42–43
   examples, 44
   in other patterns, 45, 47, 48, 49, 55
   problem, 42
   solution, 43
ordering constraints, 43
organizing principle, 60
   by data decomposition, 61
organizing principle (cont.)
  linear, 61
  recursive, 61
by flow of data, 62
  irregular, dynamic, 62
  regular, static, 62
by tasks, 61
  linear, 61
  recursive, 61
orphan processes, 277
overhead, parallel, 4, 19–21
overlapping communication and computation, 22
owner-computes filter, 138

P
page-placement algorithm. See also Nonuniform Memory Access
  first touch, 164
pairwise synchronization, in OpenMP. See OpenMP.
parallel algorithm. See also scalable algorithm
  constraints, 59
  description, 73
  design, 2, 20, 25, 29
  development, 50
  effectiveness, 4, 36
  organizing principle, 35
parallel architectures, 8–12. See also Flynn’s taxonomy
parallel computation, quantitative analysis, 18–21
parallel computers, 1
  processing elements (PEs), 17
parallel computing, 3, 13, 16–18
parallel construct, in OpenMP, 255
parallel divide-and-conquer matrix multiplication, 171. See also matrix multiplication
parallel DO construct, in OpenMP, 261
parallel file system, definition, 315
parallel for construct, in OpenMP, 255
parallel I/O, 15
parallel language definition. See
  explicitly parallel language;
  implicitly parallel language
parallel linear algebra library. See ScaLAPACK
parallel loop, 57
parallel Mandelbrot set generation, 149. See also Mandelbrot set
parallel matrix multiplication, 97. See also matrix multiplication
parallel mergesort, algorithm, 169. See also mergesort
parallel overhead, 315
parallel pipeline, 111
parallel program performance, 18–22
parallel programming, 3–4
  background, 7
  challenges, 3
  environments, 2, 12–16
  pattern languages, usage, 4–5
  support, 15
parallel programs vs operating system concurrency, 7–8
parallel region, in OpenMP, 76, 168, 169, 253
Parallel Telemetry Processor (PTEP), 73
Parallel Virtual Machine (PVM)
  capability, 220
  definition, 316
  programs, 129
parallelism. See also AND parallelism;
  fine-grained parallelism;
  incremental parallelism;
  loop-based parallelism;
  OR parallelism
  definition, 315
  strategies, 246
Parlog, 14
parsing, Recursive Data pattern example, 101–102
Partitioned Global Address Space Model, 252
pattern language
  concept, 2
  usage, 3–4
Pattern Languages of Programs (PLoP), 4–5
patterns
  Chain of Responsibility, 113
  decomposition, using, 25
  dependency analysis, 25
  Facade, 116
  format, 4
  Pipes and Filters, 112
program structuring, 6, 69
representing data structures, 123
Visitor, 4
PE. See processing element
peer-to-peer computing, definition, 315
perfect linear speedup, 19
performance
analysis tools, 153
bottlenecks, elimination, 153, 175
goals, 8
problem, 183, 187
persistent communication, 286
PET. See Positron Emission Tomography
PETsc. See Portable Extensible Toolkit for Scientific Computing
pipeline. See also parallel pipeline algorithms, 40, 103
assembly-line analogy, 104
computation, 55
draining, 105
example, three-stage pipeline, 104
elements, data flow
(representation), 107–108
filling, 105
stages, 105
defining, 106–107
usage, 110–112
Pipeline pattern, 58, 60–62, 103–114
computation, structuring, 107
context, 103
examples, 109–112
forces, 104
in other patterns, 40, 55, 64, 115, 120, 125, 291
problem, 103
related patterns, 112–114
solution, 104–109
Pipes and Filters pattern, 112
pipes, in UNIX, 7
Pixar, 1
PLAPACK, 39, 211, 215
PLoP. See Pattern Languages of Programs
pointer jumping, 99
point-to-point message passing, 277–279, 284–288
poison pill, 145
pooled threads, 294
POOMA, 215
Portable Extensible Toolkit for Scientific Computing (PETsc), 215
Portable Operating System Interface (POSIX)
definition, 315
threads (Pthreads), 185
definition, 316
Positron Emission Tomography (PET), 26
POSIX. See Portable Operating System Interface
post Hartree Fock algorithms, 211
pragma. See OpenMP
precedence graph, definition, 315
preconfigured clusters, 12
prefix scan, Recursive Data pattern
example, 101
private clause, in OpenMP, 263
private variable, 263
problem solving environments, 211, 215
processes, 16
creation/destruction, 220–221 (see also Java; MPI; OpenMP)
definition, 315
ID, 122
lightweight (see threads)
migration, definition, 315
processing element (PE), 17, 31–32, 52
availability, 50–51
data structures, sharing, 51
definition, 316
tasks, mapping, 76–77
process group, in MPI, 274
process migration, 315
program structuring patterns, 122–123
program transformations
coalescing loops, 154
merging loops, 154
semantically neutral, 155
programming environment,
definition, 316
programming model, definition, 316
fork/join, 172
Prolog, 214
PSEs. See problem solving environments
PTEP. See Parallel Telemetry Processor
Pthreads. See Portable Operating Systems Interface
public resource computing, 2
PVM.  See Parallel Virtual Machine

Q
quadratic recurrence relation, 70
Quadrics Apemille, 8
quantum chemistry, 73, 141
queue.  See block-on-empty queue;
distributed queue;
non-blocking queue; shared
queue
quicksort, 77

R
race conditions, 17–18
definition, 316
radar, 111
ratio of computation to overhead, 52, 82
rank, in MPI, 128, 136, 282
ray-tracing algorithms, 66
read/write data, 154
read/write locks, 176–177
readers/writers, 176–177
concurrency-control protocol, 181
read-only shared data, 46
read-write shared data, 47
ready communication mode,
in MPI, 287
receive operation, 89
recurrence relations, 101, 103
recursion, 74
Recursive Data pattern, 58, 61–62, 97–102
context, 97
data decomposition, 100
examples, 101–102
forces, 99
in other patterns, 79, 97, 125, 127, 168
problem, 97–99
related patterns, 102
solution, 99–101
structure, 100
synchronization, 100
recursive data structures, 35, 62, 79, 97
recursive decomposition, 79
recursive doubling, 99, 250
recursive Gaussian Quadrature,
usage, 172
recursive parallel decomposition, 195
reduction, 13, 67.  See also tree-based
reduction
definition, 316
operators, 246
implementation, distributed
results, 247
performance, 249
recursive-doubling implementation
for associative operators, 250
serial implementation for
nonassociative operators,
248–249
tree-based implementation for
associative operators, 249–250
reduction clause, in OpenMP, 265
reentrant lock, 301
refactoring, 316–317.  See also
incremental parallelism
definition, 316–317
regular decomposition, 54
relative speedup, 19
remote procedure call (RPC)
definition, 317
renderfarm, 1
replicated data, 70
request handles, 284
ring of processors, 238
RMI, 242
round-off errors, 153, 156
row-based block decomposition, 36
RPC.  See remote procedure call
runtime library, in OpenMP, 232
runtime schedule, in OpenMP, 271
S
scalable algorithm, 124, 129, 134
Scalable Simulation Framework
(SSF), 119
ScaLAPACK, 39, 78, 97, 142, 206, 211
scaled speedup, 21
schedule clause, in OpenMP, 271
schedule(dynamic), 143, 151
schedule(guided), 271
schedule(static), 87
schedule(runtime), 271
scheduling
dynamic, 69, 271
overhead, 162, 271
static, 68
strategy, 311
scientific computing, 39, 97, 126, 198

Search for Extraterrestrial Intelligence (SETI)
- radio telescope data, 151
- SETI@home project, 2

sections
- construct, in OpenMP, 262
- semantically neutral transformations, 262
- semaphore. See also counting semaphore
definition, 317
send operations, 89
separable dependencies, 69
sequential algorithm, 73, 124
running time, 100
transformation, 135
sequential code reuse, 82–83
sequential divide-and-conquer algorithms, 73–74
sequential equivalence, 84
serial computation, 20, 248–249
serial fraction, 20, 307
definition, 317
serial reductions, 250
serialization, 242

SETI. See Search for Extraterrestrial Intelligence

shadow copies, 83
shape of chunks, in domain decomposition, 79
shared address space
definition, 317
environments, 45
shared data. See also read-only shared data; read-write shared data
- accumulation, 47
- ADT, 123
effectively local, 46–47
- identification, 46
management techniques, 174
- multiple-read/single-write, 47
read-only, 46
read/write, 47

Shared Data pattern, 122–123, 173–182
context, 173
examples, 179–181
forces, 174
in other patterns, 68, 77, 154, 183, 184, 187, 196, 231

problem, 173
related patterns, 182
solution, 174–178

shared memory, 10–11
- APIs, 232
- computers, 164
definition (see also virtual shared memory)
environment, 32, 37
MIMD computers, 211
models, 15
- multiprocessor computers, 149
node, 12, 13
support, 125
system, 31
shared-memory programming
- environments, 35
model, 87
advantages, 14
shared nothing, definition, 317
shared queue, 123. See also distributed queue operations, 175

Shared Data pattern example, 179

Shared Queue pattern, 122–123, 183–198
context, 183
examples, 194–196
forces, 183
in other patterns, 107, 117, 144, 147, 148, 151, 169, 173, 174, 179, 182, 301–304
problem, 183
related patterns, 196–197
solutions, 183

shell programs. See UNIX
shotgun algorithm, 2
signal processing, 103
applications, 60
SIMD. See Single Instruction Multiple Data

simplicity, 30
simulation. See discrete-event simulation

simultaneous multithreading (SMT)
definition, 318
usage, 162

single
- construct, in OpenMP, 87, 246, 262
Single Instruction Multiple Data (SIMD)
architecture, 319
definition, 318
platform, 100–101
Single Instruction Single Data (SISD), 8
Single Program Multiple Data (SPMD)
algorithms, 129
approach, 127
definition, 318
SPMD pattern, 122–123, 125, 126–143
context, 126–127
eamples, 129–142
forces, 127–128
in other patterns, 70, 71, 72, 85, 88, 95, 97, 107, 143, 149, 152, 157, 160, 162, 167, 172, 199, 211, 220, 223, 236, 238, 239, 276
problem, 126
related patterns, 142–143
solution, 128–129
single-assignment variable
definition, 318
single-thread semantics, 212
single-threaded programs, 155
SISAL, 215
SISD. See Single Instruction Single Data
SMP. See symmetric multiprocessor
SMT. See simultaneous multithreading
software caching, 177–178, 181
sorting algorithm, 77
Space Telescope Science Institute
(STSI), 110–111
Space-Time Adaptive Processing
(STAP), 111
spawn, 220
SPEC OMP2001 benchmark, 179, 181
SPEEDES. See Synchronous Parallel Environment for Emulation and Discrete-Event Simulations
speedup. See also fixed-time speedup;
perfect linear speedup;
relative speedup; scaled speedup
definition, 318
maximum, 307
spin lock, 241
SPMD. See Single Program Multiple Data
square chunk decomposition, 82
SSF. See Scalable Simulation Framework
standard communication mode, in MPI, 286
STAP. See Space-Time Adaptive Processing
Steele, Guy, 101
static schedule, 68–69, 271
status variable, in MPI, 278
stride, 95
definition, 318
structured blocks
in OpenMP, 218
directive formats, 257–259
STSI. See Space Telescope Science Institute
suitability for target platform, 32, 49–50, 52, 59, 60
supercomputers, usage, 12
supporting structures, 211
Supporting Structures design space, 5, 121
abstraction, clarity, 123–124
efficiency, 125
environmental affinity, 125
forces, 123–125
maintainability, 124
patterns, 125–126
scalability, 124
sequential equivalence, 125
surface-to-volume effect, 82
symmetric multiprocessor (SMP), 10, 13. See also tightly coupled symmetric multiprocessors
computers, 157
definition, 318
workstations, 316
cluster, 17
symmetric tridiagonal matrix, 78
synchronization. See also memory;
OpenMP
constructs (see high-level synchronization constructs)
definition, 319
fences, 222
mechanisms, 303–304
overhead, 53
requirement, 87
usage, 39, 221–237
synchronized blocks, in Java, 233–235, 297–299
associated object, 298
association, 298–299
deficiencies, 235–236, 301
placement, 299
specification, 297
synchronous, asynchronous (contrast), 17, 50
synchronous communication mode, in MPI, 286
Synchronous Parallel Environment for Emulation and Discrete-Event Simulation (SPEEDES), 119
systolic algorithm, 319
systolic array, 103
definition, 319

T
target platform. See also suitability for target platform
considerations, 63
number of PEs, 50
number of UEs, 59
target programming environments, 216
Task Decomposition pattern, 25, 27, 29–34
context, 29
elements, 31–32
forces, 30
in other patterns, 36, 37, 38, 39, 41, 42, 44, 47, 49, 51, 54, 63, 64, 134
problem, 29
solution, 30–31
task groups
asynchronous/synchronous interaction, 55
hierarchical, 55
temporal constraints, 39, 47, 49, 54, 56
usage, 44
task migration, 119
Task Parallelism pattern, 58, 60–61, 63, 64–73
collection idioms, 70
context, 64
dependencies, 66
elements, 70–73
forces, 65
in other patterns, 79, 80, 82, 97, 107, 109, 112, 114, 125, 126, 127, 136, 140, 143, 146, 149, 153, 154, 162, 167, 173, 174, 181, 182, 311, 319
problem, 64
program structure, 69–70
schedule, 68–69
solution, 65–70
tasks, 65–66
task queue, 70. See also double-ended task queue; master/worker algorithms
definition, 319
initialization, 147
OpenMP, 77, 169
task-based decomposition, 27, 29, 46
production, 32
usage, 34
task-local data, 45
sets, 46
task-parallel computations, sequential composition, 66
task-parallel problems, 69, 140, 162
tasks
collection, 42
constraints, 42–43
data sharing, 51
definition, 26, 319
distribution, 68
graph, 74
grouping, 39–42, 55
identifying, 36
mapping, 76–77 (see also processing element; unit of execution)
migration, 119
ordering constraints, 43
organization, 60, 63
regularity, 54
restructuring, 46
simultaneity, 124
synchronous/asynchronous interaction, 50, 55
TCGMSG, 73, 152
TCP/IP
socket, 242
support in Java (see Java)
temporal dependencies, 42
termination condition, 70
termination detection algorithms, 146
Thinking Machines, 8
Thread.currentThread, usage, 193
thread pools
thread-pool-based Fork/Join implementation (see Fork/Join pattern)
ThreadPoolExecutor class, usage, 148
threads, 16. See also master thread;
POSIX threads
creation (see Java; MPI; OpenMP)
definition, 319
destruction (see Java; MPI; OpenMP)
fork, 122, 150–151
IDs, 123, 157
finding, 133
referencing, 157
usage, 122 (see also SPMD pattern)
instances, 218
management, 76
team of threads 162, 168, 253
termination, 186 (see also child thread)
thread safety, 220
thread visible data, 241
three-stage pipeline, example, 109
throughput, 109
tightly coupled symmetric multiprocessors, 8
time-critical loops, 129
timeout, 300
time stamps, 118
time-stepping methodology, 133
time warp, 118
Titanium, 16, 252
Top 500 list, 127
top-level task, 77
Toy Story (1995), 1
tradeoff, total work for decrease in execution time, 99
transaction, 308
transpose algorithm, 53
Transputer, definition, 319
trapezoid rule for numerical integration, usage, 129, 159
tree-based reduction, 249
tridiagonal linear systems. See symmetric tridiagonal matrix
try-catch block, in Java, 302
tuple space. See also Linda
definition, 319
two-dimensional (2D) block distribution, 200
two-dimensional (2D) Fourier transform computations, 111–112
two-sided communication, 251
U
UE, See unit of execution
unit of execution (UE) 16. See also lightweight UE; processes; threads
assignment, 152
communication impact, 237–251
definition, 320
identifier, 128, 200, 207
management, 217–221
mapping, 200–201 (see also direct task/UE mapping; indirect task/UE mapping)
number
target architecture implications, 51, 59
tasks, mapping, 76–77
round robin assignment, 200
UNIX
context, 319
pipes, 7
shell programs, 7
Unified Parallel C (UPC), 252
util.concurrent package, in Java, 293, 294
V
variable scope in OpenMP. See
firstprivate; lastprivate; private
vector data, 9
vector processing, 103
vector processor, 8
vector supercomputer, definition, 320
virtual distributed shared memory systems, 12
virtual machines. See Java Virtual Machine; Parallel Virtual Machine
virtual shared memory, definition, 320
Visitor pattern, 4
Vlissides, John. See Gang of Four
volatile variable, in Java, 225
von Neumann model, 8, 12

W
wait and notify, 299–301
wait set, 299
war gaming exercises, 119
WESDYN molecular dynamics program, 34
whole genome shotgun algorithm, 2
WOMPAT. See Workshop on OpenMP Applications and Tools

WOMPEI. See Workshop on OpenMP Experiences and Implementations
work stealing, 69
worksharing constructs in OpenMP. See loop-level worksharing constructs; OpenMP
Workshop on OpenMP Applications and Tools (WOMPAT), 15, 166
Workshop on OpenMP Experiences and Implementations (WOMPEI), 166
workstation. See also multiprocessor workstations
cluster, 108
farm, definition, 320
networks, 52