# Index

Page numbers followed by "*f*" indicates figures and "*t*" indicates tables.

**A**

Absolute speedup, 56
Acoustic wave equation, 266
Advanced Vector Extensions (AVX), 45
`affinity_partitioner`, 352
Algorithm strategy patterns, 80
  programming, 19, 20
Algorithm's span, 4, 5
Algorithmic skeletons, 1, 20
Aliased inputs, 87
Aliasing, 96
Amdahl's Law, 5, 52, 59–60, 64, 254
AoS, *see* Array of structures
Application programming interface (API), 25, 32, 346
ArBB, *see* Array Building Blocks
Arithmetic intensity, 50, 74, 204, 250, 265
  impact of caches on, 267–270
  of SAXPY, 125
  raising with space–time tiling, 270–272
Array Building Blocks (ArBB), 23, 25, 27, 97, 115–116
  automatic fusion in, 153–154
  dot product in, 161–162
  using elemental functions, 129–130
  features of, 34–35
  implementation of, 152
  implementation of stencil, 275–277
  integration, 175–177
  of Mandelbrot, 134–137
  scan pattern, 165
  use of, 26, 37
  using vector operations, 128–129
Array expressions, 27
Array indices, 96
Array notation, 338
  Cilk Plus with, 127–128, 134
  array sections
    avoid partial overlap of, 342–343
    operations on, 340–341
    reductions on, 341–342
    specifying, 339–340
  statements, 285
Array of structures (AoS), 194–195, 196*f*
Array section operator, 339, 340
Associative array, 105
Associative operations, 102, 106, 108, 185, 261, 286, 335
Asymptotic complexity, 65–67
Asymptotic speedup and efficiency, 67
`atomic`, 356–358
Atomic operations, 73, 112, 349, 356–357
Atomic scatter pattern, 183–184, 183*f*
Attached devices, 48–49
`auto_partitioner`, 352
Auto-vectorization, 349
Automatic fusion in ArBB, 153–154
Autoparallelization, 14
Autotuning, 50
AVX, *see* Advanced Vector Extensions

**B**

Bandwidth, 10, 46, 99, 139, 179, 190, 204, 227, 319
Barriers, 33, 88, 115
Basic Linear Algebra Subprograms (BLAS), 124, 125
  column-major layout, 315
Big O notation, 66
Big Omega notation, 66
Big Theta notation, 66
Bilateral filtering, 200
Bin pattern, 189, 189*f*
Binary search, 302
`bindex_type`, 309
Binomial lattice recurrence, 246, 246*f*
Bins
  choosing, 309
  code for mapping keys to, 310
BLAS, *see* Basic Linear Algebra Subprograms
`blocked_range`, 351–352
Blocking, *see* Tiling
Bound-to-item approach, 256
Bound-to-stage approach, 256
Branch and bound pattern, 41, 103, 108–109, 114
Brent's Lemma, 63–65
Built-in function, 342
Burdened span, 63
Burrows–Wheeler transform, 291, 292
By reference, 363, 364
By value, 363, 364
Bzip2
  algorithm, 291–292
  three-stage pipeline for, 293*f*

**C**

C language, example of, 87
C syntax, 337–338

C++ compilers, 125
C++ syntax, 335–337
C++11, 358–359, 361
  features, 358
  lambda expressions, 313, 345, 361, 365
  `std::move`, 365
C++98 equivalent, 361
Cilk++, 346–347
Cache, 5, 44, 46, 73
  direct-mapped cache, 46
  impact on arithmetic intensity, 267–270
  set-associative cache, 46
  tiling stencils for, 202–203, 203*f*
Cache fusion, 140–141, 140*f*
Cache lines, 46, 202
Cache locality, 227–230
Cache oblivious algorithms, 50, 73, 209, 227–230, 273, 315
`call` function, 129
Capture
  by reference, 363, 364
  by value, 363, 364
Category reduction, 92, 106–107, 107*f*, 189
Centroid of cluster, 279
Cholesky decomposition, 317–318, 322*f*
Cilk Plus, 19, 22, 23, 25, 27, 28, 112–113, 125, 311, 349, 360
  array notation, 17, 224
  binomial lattice in, 250
  cache fusion, 141
  code, 272–273, 275
  dot product in, 158–159
  explicit fusion in, 153
  features of, 29–30, 245
  for fork–join, 212–213, 213*f*
  history, 346–347
  hyperobjects, 239, 240*f*, 331
  implementation of, 152
  implementation of SAXPY operation, 127
  integrated table preparation in, 173
  for integration, 170
  k-means algorithm with, 281–284
  Karatsuba multiplication in, 225
  Mandelbrot in, 132–134
  map pattern in, 215–217, 216*f*
  parallel merge in, 301, 304
  parallel reduction in, 238
  parallel scan in, 244
  parallel symmetric rank update in, 321
  parallel triangular solve in, 320
  pipeline in, 258–261, 259, 261
  predefined holders, 335
  quicksort in TBB achieves, 237
  reduction operation in, 18
  repacking and subsorting using, 312
  scan pattern, 164–165
  semi-recursive parallel Quicksort using, 234
  setup, 346
  unique characteristics, 329–331, 330*f*
  use of, 36
  using local reducer in, 241
  vectorized function for addition in, 26
Cilk Quicksort, 231–233
`cilk_for`, 30, 113, 158
`cilk_for` constructs, 127, 134, 332–333
`cilk_forc`, 213, 215
`cilk_scan`, 165
`cilk_spawn` keyword, 19, 20, 112, 212, 333–334
`cilk_sync`, 30, 212, 333–334
Class comparator, 362
Class `View`, 283
Closures, 87, 97, 118
Cluster, 52, 101
  centroid of, 279
Code fusion, 139–140, 139*f*
Collective operations, 116, 145
Collisions, 102, 106, 183
Combiner function, 90, 145
Comma operator, 340
Communication, 39, 50, 73, 121
  optimizing stencils for, 203–204
Commutative operations, 107, 185, 286
Commutativity of reduction, 148, 149*f*
Compare-and-swap loop, 357–358
Composability, 23, 27–28, 329
Compositional pattern, 81*f*
Computer architecture, 325–326
Concrete instantiation of test function, 171
Concurrency, 67
Concurrent Clean, 108
Condition variable, 359
Consumer, 261
Continuation, 219
  passing style, 221, 233
  passing tasking, 234
Control dependency, 39
Control flow, fork–join, 210*f*
CPU2006 SPEC benchmarks, 14, 16*f*
CUDA, 53
Cumulation, 169
Cyclic redundancy check (CRC) code, 291

## D

DAG model of pipeline, 256*f*
Data, 39
Data amplification, 191

Data dependencies, 83, 84, 86, 102
  graph, 246
Data layout options for SoA and AoS, 196*f*
Data locality, 47, 50–51, 179, 182
Data organization
  AoS, 194
  SoA, 195
Data parallelism, 20, 40, 81, 359
  expression of regular, 24–27
Data-dependent control flow, 136
Data-intensive applications, 179
Deadlocks, 72, 113, 254
Dependency, 39
Design patterns programming, 19
Determinism, 3, 29
Deterministic map pattern, 124
`deterministic_parallel_reduce` function, 114
Deterministic patterns, 79, 179
Deterministic reduction, 151
Diamond-shaped subgraph
  decomposing, 247, 247*f*
  parameters for, 248, 248*f*
Direct memory access (DMA), 48
Directed acyclic graph (DAG), 62, 107
Divide steps, 281
Divide-and-conquer algorithms, 179, 191, 209, 210
  base cases selection of, 217–218
  complexity of parallel, 221–223
  implementation of, 234
  nested fork–join control flow in, 211*f*
Divide-and-conquer pattern, 142, 317
Divide-and-conquer strategy, 20
DMA, *see* Direct memory access
`do-while` loop, 281
Dot product, 154
  ArBB, 161–162
  in Cilk Plus, 158–159
  implementation of
    OpenMP, 160–161
    serial, 154–155
  in SSE intrinsics, 155–156
  in TBB, 156–158
Double-data-rate (DDR) signaling, 10
Dwarfs, 325

## E
Efficiency, 56
Elastic wave equation, 266
Elemental functions, 22, 25, 27, 88, 121, 123, 181*f*, 191, 332, 344
  ArBB using, 129–130
  attribute syntax, 345
  computation in ArBB, 35
  Mandelbrot, 136
`elementwise_reducer`, 283, 284
Embarrassing parallelism, 20, 121, 145
`empty_task`, 233
`enumerable_thread_specific`, 354, 358
Expand pattern, 106, 106*f*, 117, 118, 179, 189, 190, 190*f*
Explicit fusion
  in Cilk Plus, 153
  in TBB, 152
Explicit parallel programming, 14–19

## F
False sharing, 47, 179, 186, 202
Fibers, 42
Fibonacci computation in C, 31
Fibonacci numbers, 30
Field-programmable gate arrays (FPGAs), 36
Fine-grain locking, 73
Flat algorithm for polynomial multiplication, 224
Flat fork–join pattern, 251
Floating-point arithmetic operations, 269, 319
`floor_lg2`, 309
Flynn's characterization, 51–52
Fold, 92
`foo`, 18
`for` loop, 123
  demonstration of, 86
  iteration using, 85
`for` statement, 4, 27
Fork–join model, 349
Fork–join pattern, 20, 39, 88, 103, 112–113, 115, 141, 142, 191, 207, 315
  for Cholesky decomposition, 322
  Cilk Plus for, 212–213, 213*f*
  control flow, 211*f*
  definition of, 210–211
  flat, 251
  implementing scan with, 241–246
  OpenMP for, 214–215
  programming model for, 211–212
  to recurrences, applying, 246–250
    analysis, 250–251
  recursive, 234
  TBB for, 213–214
Fortran rules, 315–317
Forward model, 266
Four-stage pipeline, using TBB, 296
Fragmented memory, 96
Free monoid, 259
Function object, 362
Functional decomposition, 40

Functors, 213
Fusing map
  and pack pattern, 189–191, 190*f*
  and reduction, 152
  and scan, 166–168
Futures pattern, 102–103

## G
Gather pattern, 101, 101*f*, 113, 115–117, 179, 180, 181*f*
  conversion to scatter pattern, 186–187
Gaussian elimination, 124
General gather pattern, 180
Generic test function for integration, 171
Geometric decomposition, 100–101, 100*f*, 115, 118, 179, 191–193
Geometry shaders, 191
Ghost cells, 203–204
Global memory, 117
`goto`, 20
Granularity, 7, 228
Graphics Processing Units (GPUs), 53
Gustafson-Barsis' Law, 60–62

## H
Hardware threads, 5, 41
Hardware trends encouraging parallelism, 7–11
Hash table, 109
Header files, 337, 346
Heap allocation, 96–97
Heterogeneous computing, 53
High-performance computing (HPC), 32, 33
Host processor, 54
HPC, *see* High-performance computing
Huffman coding, 292
Hyperobjects, 112, 238–241, 281–285, 331, 334–338

## I
Idioms, 80
`if` statement, 27
ILP wall, *see* Instruction-level parallelism wall
Implicit Index, 342
Instruction parallelism, 44–45
Instruction-level parallelism (ILP) wall, 8–11
Integration, 169–170
  ArBB, 175–177
  Cilk Plus for, 170
  OpenMP, 170–172
  serial implementation, 170
  TBB, 172–175
Intel 4004 4-bit microprocessor, 11
Intel Cilk Plus, 22, 347
Intel Concurrent Collections, 322
Intel TBB, *see* Threading Building Blocks
Intel Xeon E7-8870 processor, 11
Intrinsics, 1
Irregular parallelism, 41
Iteration pattern, 84–87

## K
k-means algorithm, 279–281, 280*f*
  with Cilk Plus, 281–284
  with TBB, 285–289
Karatsuba multiplication of polynomials, 224–227
Kernels, 117, 130, 138
Keyword spelling, 332
Knights Corner, 11

## L
L-system substitution, 190
Lambda expression, 125, 213, 285, 310, 361–365
  C++11, 313
Lambda functions, 97, 107
Lanes, 42
Last in first out (LIFO), 96
Latency, 10, 46, 53, 55, 104, 254
  hiding, 56, 204
LIFO, *see* Last in first out
Linear Algebra Package (LAPACK), 315
Linear pipeline, 99
Linear speedup, 56
Linked list, 97
Little's formula, 67–68
Lloyd's algorithm, *see* k-means algorithm
Load balancing, 7, 132, 218–221
Load imbalance, 74
Locality, 5
  lack of, 73–74
Lock, 70, 72, 73
Loop fusion, 153
Loop-carried dependencies, 85

## M
Machine model, 44–49
  attached devices, 48–49
  evolution of, 53–54
  Flynn's characterization, 51–52
  instruction parallelism, 44–45
  key features for performance, 50–51
    data locality, 50–51
    parallel slack, 51
  memory hierarchy, 45–47

multiprocessor systems, 48
virtual memory, 47–48
Mainstream computer architectures, 2
Mandatory parallelism, 261–262
`mandel` function, 134
Mandelbrot set
ArBB implementation, 134–137
Cilk Plus implementation of, 132–134
description of problem, 131–132
OpenCL implementation of, 138–139, 138*f*
serial implementation of, 132, 133
Many Integrated Cores (MIC), 45, 49, 54
Many-core co-processor, 53
Many-core processors, 2
Map of sequence operations, 139–141
`map` operations, 129, 276
Map pattern, 88–89, 113–115, 117, 121, 123–124, 145, 199, 209, 253, 351
recursive implementation of, 215–217, 216*f*, 238
related patterns, 141–142
serial and parallel execution of, 122*f*
MapReduce, 122
Masking, 42–43
Matrix multiplication, 319–321
Matrix of buckets, 310
Member functions, 97
Memory bandwidth, 139, 227, 303, 312, 319
Memory fences , 69
Memory hierarchy, 45–47, 209
Memory operations, 269
Memory subsystem, 96, 202
Memory wall, 8
effects of, 10
Merge scatter pattern, 179, 183, 184*f*, 185
Merge sort
parallelizing, 299–301
TBB parallel merge, 301
work and span of parallel merge, 301–303
MIC, *see* Many Integrated Cores
Microprocessors, 11
Microsofts Parallel Patterns Library (PPL), 360
MIMD, *see* Multiple Instruction, Multiple Data
Mixed capture modes, 364
`Monoid`, 283
Moore's Law, 7, 8*f*, 10, 11, 13
Motifs, 80, 325
Move-to-front coding, 292
Multicore era, 11, 13
performance in, 15*f*
Multicore processor, 11, 44, 45*f*
Multiple Instruction, Multiple Data (MIMD), 52
`MultiplyAdd`, 229
Multiply-nested loop, 205*f*
Multiprocessor systems, 48
Mutex, 71, 73, 358, 359
Mutual exclusion, 71
`my_simple_add`, 26

## N
`neighbor` function, 142
Nested fork–join control flow, 211*f*
Nested parallelism, 23
Nesting, 20, 80–82, 81*f*, 112–113, 193
Network Interface Controller (NIC), 48
Non-associative operation, 147
Non-determinism, 29
Non-deterministic patterns, 108–109, 184
Non-deterministic program testing tools, 29
Non-scalar operands, 340
Non-uniform memory access (NUMA), 48

## O
Objects, 97–98
Off-chip memory, 10
Offline program analysis, 9
Offload, 53
On-chip local memory, 117
One-dimensional stencil, 272*f*
Online algorithm, 253
OpenCL, 23–25, 28, 53
features of, 35–36
implementation of, 151
implementation of SAXPY, 130–131
Mandelbrot computation for, 138–139, 138*f*
programming model, 116–118
recursive divide-and-conquer in, 142
use of, 37
OpenMP, 24, 28
cache fusion, 140
directives, 14
features of, 32–34
for fork–join, 214–215
implementation of, 152
dot product, 160–161
for integration, 170–172
interface, 114–115
Mandelbrot in, 134, 136
recursive divide-and-conquer pattern, 142
SAXPY in, 128
scan pattern, 165–166
stealing semantics for, 221
use of, 37
`operator+=`, 281
Optimization of map, 153*f*, 169*f*

Optional parallelism, 23, 261–262
Over-sampling, 309
Overhead, 74–75

## P

Pack pattern, 98–99, 98*f* 116, 117, 179, 187–189, 187*f*, 245
  fusing map and, 189–191, 190*f*
Packing, 43
Page fault rate, 47
Parallel algorithm
  and patterns, 325
  designing, 5
Parallel computation, 29, 39
Parallel control patterns
  fork–join pattern, 88
  map pattern, 88–89, 89*f*
  recurrence pattern, 95
  reduction pattern, 90–92, 91*f*
  scan pattern, 92–95
  stencil pattern, 89, 90*f*
Parallel data management patterns, 98–102
`parallel_deterministic_reduce`, 151, 354
`parallel_do` construct, 114
`parallel_for` functions, 4, 114, 352
`parallel_foreach` functions, 114
`parallel_invoke`, 213, 229, 354
Parallel merge, 299–301
  sort, 303–304
  TBB, 301
  work and span of, 301–304
Parallel models, comparison of, 141
Parallel partitioning, 311
Parallel patterns, 21*f*, 79
`parallel_pipeline` construct, 114, 354
Parallel pipeline stages, 254–255
Parallel programming, 39, 349
  machine model, 44–49
    attached devices, 48–49
    instruction parallelism, 44
    memory hierarchy, 45–47
    multiprocessor systems, 48
    virtual memory, 47–48
  mechanisms, 41–44
    thread parallelism, 41, 42
    vector parallelism, 41, 42
  performance theory, 54–68
  pitfalls, 68–75
  strategies for, 40–41
  vocabulary and notation, 39–40
Parallel programming models, 3, 21, 22*f*, 326–327, 329
  abstractions, 23–24
  composability, 27–28
  expression of regular data parallelism, 24–27
  portability of functionality, 28
  properties, 21–22
  safety, determinism, and maintainability, 29
Parallel Random Access Machine (PRAM), 48
`parallel_reduce` construct, 114, 151, 156, 157
Parallel scan, 94*f*, 242, 242*f*
  algorithm, 241
  top-level code for tiled, 243
  upsweep and downsweep phase, 244
`parallel_scan` construct, 114, 165, 172
Parallel slack, 51
Parallel systems, 325–326
Parallel task, 255
Parallel thinking, 2–4
Parallelism, 2, 3, 209, 350, 359
  hardware trends, 7–11
  historical trends in, 11–14
Parallelization, 1, 9, 41, 50, 145, 179
Parallelized map, 122
Parallelizing algorithms, 85, 86
Parameterized code, 50
Partial differential equations (PDEs), 95, 201, 203
Partition pattern, 100, 100*f*, 191–193, 191*f*, 192*f*
Partitioners, 352–353
Partitioning, 307, 309–310
  and repack/subsort phase, 312
Partitioning data, 179
Pattern, 39, 79
  programming model supports, 110–112*t*
    array building blocks, 115–116
    Cilk Plus, 112–113
    OpenCL programming model, 116–118
    OpenMP interface, 114–115
Pattern-based programming, structured, 19–20
PDEs, *see* Partial differential equations
Performance of parellel programming, 4–7
Performance portability, 28
Performance theory, 54
  Amdahl's Law, 59–60
  asymptotic complexity, 65–67
  asymptotic speedup and efficiency, 67
  efficiency, 56
  Gustafson-Barsis' Law, 60–62
  latency and throughput, 55–56
  Little's formula, 67–68
  power, 57–58
  scalability, 57
  speedup, 56–57
  work-span model, 62–65
Permutation scatter pattern, 116, 183–185, 183*f*
Pervasive parallelsim, 7
Pipeline, 99–100, 99*f*, 114
  basic, 253–254
  four-stage, 296

general topologies, 261
implementation of, 255–257
mandatory versus optional parallelism, 261–262
with parallel stages, 254–255
programming model supporting for, 257–261
three-stage
using Cilk Plus, 296–297
using TBB, 292–296
Pitfalls, 68–75
deadlock, 72
lack of locality, 73–74
load imbalance, 74
lock, 70
mutual exclusion, 70
overhead, 74–75
race condition, 68–70
strangled scaling, 73
Pochoir stencil compilers, 273
Point-to-point synchronization, 33
Pointers, 86, 96
Polyhedral theory, 206
Polynomials, Karatsuba multiplication of, 224–227
Portability, 23
of functionality, 28
Potential parallelism, 51, 81, 209
Power, 57–58
Power wall, 8, 9, 11
PPL, *see* Microsofts Parallel Patterns Library
Pragma, 53, 128
Pragma markup, 211
`#pragma simd`, 343–344
PRAM, *see* Parallel Random Access Machine
Priority scatter pattern, 183, 184*f*, 185–186
Producer, 261
Producer–consumer relationship, 99
Programmer, goal of, 2
Programming model
for fork–join, 211–212
supporting for pipelines, 257–261
supports pattern, 110–112*t*
Array building blocks, 115–116
Cilk plus, 112–113
OpenCL programming model, 116–118
OpenMP interface, 114–115
TBB, 113–114

## Q
Quadratic function, 131
Quicksort, 105, 218, 230–231, 299, 307, 313
Cilk, 231–233
TBB, 233–237
work and span for, 237–238
`quicksort_task`, 233, 235

## R
Race conditions, 68–70, 113, 179, 191
Random read and write, 96, 116
Reassign steps, 281
Recurrence pattern, 95, 113, 199, 204–207, 205*f*, 206*f*
Recurrence relations, 299
Recurrences
fork–join to, 246–250
for work, 302
and span, 304
Recursion, 87–88, 112, 209
Recursive Cholesky decomposition, 317–318
Recursive divide-and-conquer pattern, 142
Recursive fork–join, 234
Recursive linear algebra, 315
Recursive matrix multiplication, pseudocode for, 229
Recursive merge, 300
Recursive pattern, pseudocode for, 247
`recursive_map`, 215
`recycle_to_reexecute()`, 237
`reducer_basic_string`, 240
Reducers, 158, 239, 283, 334–338
in Cilk Plus, 241
Reduction operations, 341
Reduction pattern, 27, 90–92, 112, 114–117, 145–146, 146*f*
fusing map and, 152
implementation of, 151–152
precision, 150–151
reordering computations, 146–148, 149, 150, 150*f*
tiling, 149–150
vectorization, 148–149
Reductions, 7, 238–241
using thread local storage for, 286
Refactoring, 26
Reflection seismology survey, 265
Registers, 45
Regular parallelism, 41
Relative speedup, 56
Reordering computations, 146–148
Repacking, 307
and subsorting, 310–312
Reverse model, 266
Reverse time migration (RTM), 265
grids in, 267
Rotate pattern, 182
RTM, *see* Reverse time migration
Run-length encoding, 291, 292

## S
Sample sort, 308*f*
parallel, top-level code for, 308
partitions, 307

Sample sort (*continued*)
  performance analysis, 312–313
  structure, 307–308
  subphases, 310–311
Scalability, 2, 56–57
Scalable parallelism, 40
Scalar function for addition in C, 26
Scalar operands, 340
Scaled vector addition (SAXPY)
  ArBB using vector operations, 128–129
  mathematical definition of, 124
  OpenCL implementation, 130–131
  serial implementation, 125
Scan pattern, 27, 92–95, 114, 116, 117, 162–164, 163*f*, 201
  ArBB, 165
  Cilk Plus, 164–165
  fusing map and, 166–168
  OpenMP, 165–166
  TBB, 165
Scatter pattern, 101–102, 102*f*, 113, 115–117, 124, 179, 182–183, 183*f*
  conversion to gather pattern, 186–187
Scattered allocations, 96
Scratch space, allocation of, 227
Search pattern, 105
`__sec_reduce_add` function, 134
Segmentation, 105, 192, 193*f*
Seismic Duck program, 265
Selection pattern, 84
Semantics, 80
  of map pattern, 121
Separating hyperplane, 204
Sequence of map operations, 139–141
Sequence pattern, 82–84
Sequential consistency, 69
Sequential semantics, 3
Sequentially consistent programs, 29
Serial computation, 29
Serial data management patterns, 95–98
Serial elision, 30
Serial illusion, 2
Serial integrated table preparation in C, 171
Serial merge sort, 300, 304
  properties, 299
Serial pattern, 79
Serial pipeline stage, 253
Serial processor illusion, 10
Serial Quicksort, 231
Serial reduction in C++, 146, 147
Serial scan pattern, 93*f*
Serial task, 255
Serial traps, 3, 14
Serial–parallel–serial pipeline, 255*f*
Serialization, 2, 3, 248
Shared design principles, with TBB, 329
Shared memory machine model, 5
Shared memory parallel programming models, 22
Shift, 179, 181*f*, 182
  implementing stencil with, 201, 202*f*
SIMD, *see* Single Instruction, Multiple Data
`simple_partitioner`, 352
SIMT, *see* Single Instruction, Multiple Threads
Single Instruction, Multiple Data (SIMD), 52
  machine, 105, 189
Single Instruction, Multiple Threads (SIMT), 52
  machine model, 104
Single Instruction, Single Data (SISD), 52
Smith–Waterman algorithm, 207, 246
SoA, *see* Structures of arrays
Software thread, 41
Sorting, 107
Space–time tiling, 95, 270–272
Space-time geometry, 271
Span, 4, 5
Speculative selection, 84, 104–105, 104*f*, 114
Speedup, 5, 41, 56–57, 209
Split operation, *see* Packing
Split pattern, 187–188, 188*f*
SSE intrinsics, *see* Streaming SIMD Extensions intrinsics
Stack allocation, 96
`std::count`, 362
`std::move`, 365
Steal continuation
  semantics, 221, 233
  vs. steal child, 219, 220*f*
Stencil operations, 101
Stencil pattern, 89, 100, 115, 118, 141–142, 179, 192, 199
  computation, 266–267, 267*f*
  implementing with shift, 201, 202*f*
  optimizing for communication, 203–204
  serial implementation of, 200
  tiling for cache, 202–203, 203*f*
Strand, 239
Strangled scaling, 73
Stream compaction, 42
Streaming SIMD Extensions (SSE) intrinsics, 45, 155–156
Strip-mining, 202
Structured parallel programming, 81
Structured serial control flow patterns
  iteration pattern, 84–87, 85*f*
  recursion pattern, 87–88
  selection pattern, 84, 84*f*
  sequence pattern, 82–84, 83*f*

Structured serial programming, 81
Structures of arrays (SoA), 194–195, 196*f*
Subscript operator, 340
Subsorting, 307
  repacking and, 310–312
Successor function, 92
`sum_and_count` objects, 281
Sum steps, 281
Summed area tables, 169
Superlinear speedup, 57
Superscalar instruction issue, 9
Superscalar sequence, 84, 102, 103*f*, 118
Swap expression, 365
Swap file, 47
Symmetric rank update, 317, 319–321

## T
Tail call, 216, 217
Tail recursion, 87
Task, 39, 44
Task blocks, 80
Task cancellation, 103, 104, 114
Task Parallel Library (TPL), 360
Task parallelism, 41
`task_group`, 355
Taxicab geometry, 271
TBB, *see* Threading Building Blocks
Term graph rewriting, 107–108
Thread parallel, 42
Thread parallelism, 41, 42
Thread-local storage, 285, 331
Threading Building Blocks (TBB), 22, 27, 113, 209, 254, 310, 349, 359, 360
  borrowing components from, 330*f*, 331–332
  cache fusion, 141
  dot product in, 156–158
  equivalent, 311
  explicit fusion in, 152
  features of, 31–32
  for fork–join, 213–214
  four-stage pipeline using, 296
  history of, 359–360
  implementation of, 151, 245
  integration, 172–175
  k-means algorithm with, 285–289
  Mandelbrot in, 132, 133
  overview of, 350*f*
  parallel merge, 301
  pipeline in, 257–258, 262
  quicksort, 233–237
  SAXPY in, 125–127
  scan pattern, 165
  shared design principles with, 329
  `task`, 355–356
    `empty_task`, 356
  three-stage pipeline using, 292–296
  unique characteristics of, 349–350
  use of, 36, 350
Threads, 3, 81, 211, 359
Three walls, 8
Three-stage pipeline using Cilk Plus, 296–297
Throughput, 55
  of pipeline, 254
Tiles, 140
Tiling, 51, 149–150, 207, 265
  space–time, 270–272
Time-varying acoustic signal, 266
TLB, *see* Translation Lookaside Buffer
TPL, *see* Task Parallel Library
Transaction pattern, 109, 185
Translation Lookaside Buffer (TLB), 47, 202
Transpose arguments, 316
Trapezoidal space–time region, 272*f*
Tree reduction pattern, 90–92, 92*f*
Triangular solve, 317–319
Two-stage pipeline, 261

## U
Uniform data, 124
Uniform parameters, 124, 125
Unpack pattern, 99, 187, 188*f*
Unparallelizable algorithm, 169
Unsplit pattern, 188, 188*f*
Unzip pattern, 180, 181*f*, 182

## V
Varying data, 124
Varying parameters, 124, 125
Vector computation in ArBB, 34
Vector intrinsics, 43
Vector parallelism, 41, 42, 218
Vector register, 42
Vectorization, 13, 43, 96, 148–149, 149*f*, 179, 192, 248, 343
  of data, 201
Vectorized function for addition in Cilk Plus, 26
Vectorized map, 122
Very Large Instruction Word (VLIW) process, 9
Virtual memory, 47–48, 96
Vocabulary and notation, 39–40
  data, 39
  dependency, 39
  fork–join pattern, 39
  task, 39

**W**

Wavefield models, 266
`while` loop
  demonstration of, 86
  iteration using, 85
Work groups, 117
Work item, 117
Work stealing, 218
Work-span bounds, 244
Work-span model, 5, 62–65
Workpile pattern, 105, 113–115, 118, 142

**Z**

Zip pattern, 180, 181*f*, 182