INDEX

A
aborts, 83
adaptive locking protocol, 466
adaptive scheduling, 422
algorithmic skeleton, 263
algorithm view, 35
allocation, 418
all-or-nothing transaction, 167
Amdahl’s law, 372
analytical model, 442
application-centric models, 44
array, 67
Array-OL, 146
atomicity, 84, 326
automated parallelization, 233
auto-tuning, 194–195
b
benchmarks, 118, 314, 386, 425
block access, 4
BlockLib, 139
Brook, 145, 201
bus interconnection, 431
C
CABAC, 282
cache, 72, 237
cache behavior, 95
cache coherence, 348
cascading aborts, 172
CAVLC, 282
Cell/B.E. processor, 39
Cell Superscalar, 43
Charm++, 45
Cilk, 37
cloud computing, 452
cluster, 113, 311
code optimization, 292
Programming Multicore and Many-core Computing Systems,
© 2017 John Wiley & Sons, Inc. Published 2017 by John Wiley & Sons, Inc.
481
code profiling, 291
collector, 268
combinability, 62
communication, 36
communication congestion, 431
communication links, 444
commutativity, 177
compiler, 109, 388
computational accelerators, 407
computing performance, 3
concurrency, 36, 228, 232
concurrent code, 81
concurrent conflicting transactions, 83
concurrent data structures, 59
concurrent program, 325
concurrent programming, 465
ConcurrentTesting, 340
concurrent transactions, 91, 173
conflicts, 94
connectivity, 237
consensus problem, 62
consistency, 84
containers, 125
contention manager, 91
context switch, 345
Corey, 457
CPU cycles, 352
CPU-intensive application, 346
crossover operator, 303
CUDA, 41, 302
CUDDP, 138
CUFFT library, 108
cyclical dependency deadlock, 328

data block, 390
data compression, 386
data exchanges, 133
data layout, 35
data locality, 431
data parallelism, 186
data-parallel skeletons, 122
data transfers, 36, 133
deadlocks, 166
debugging, 341
decoder, 284
decompressions, 388
deque, 66
desk checking, 330
diagnostic tools, 358
dictionary, 68
dining philosopher, 166
distributed desk checking, 331
distributed programming, 451
distributed real-time applications, 464
distributive review, 325
DOACROSS parallelism, 205
DOALL parallelism, 205
dynamic data structure, 218
dynamic scheduling, 421

E
eager update, 88, 207
efficiency, 432
elastic transactions, 179
embarrassingly parallel applications, 13
emitter, 268
encoder, 284
energy consumption, 352
evolutionary algorithms, 301
execution flow, 420
execution plan, 130
execution time, 432

F
fairness, 353, 466
false conflicts, 94, 176
farm paradigm, 271
FastFlow, 148, 262
fat-pointer, 220
Flynn’s taxonomy, 5
fragmentation, 69
frame level, 286
functional-level parallelism, 287
functional parallelism, 4

G
garbage collection, 70
GenerOS, 457
genetic programming, 302
global scheduler, 463
global scheduling, 463
GPGPU, 301
GPMCs, 39
GPUs, 39
granularity, 466
graphics processor, 72
Grid, 452
Grid middleware, 460
Grid OS, 460
Grid systems, 460

H
hash table, 68
H.264/AVC, 282
heterogeneous architectures, 101
heterogeneous multicores, 19
high-performance computing, 431, 451
H.264/MPEG-4, 281
homogeneous components, 411

I
ILP wall, 11
instruction set architecture, 238
Intel TBB, 39
interconflicts, 94
intercore communication, 156
interleaving, 333
intraconflicts, 94
invisible reads, 89
invisible read transactions, 174
I/O latency, 357
island model, 306
isolated parallel program, 336
isolation, 84
iterator, 150

J
Java, 169
JavaGrande, 200
J2EE, 363
joining, 191

K
kernel, 345
kernel function, 132

L
LAMP stack, 365
latency, 264, 369, 465, 466
layered design, 262
lazy update, 88, 207
linearizability, 60, 86
linked lists, 67
Linux, 170, 453
Linux kernel, 465
list, 67
load balancing, 73
local data structures, 466
lock-free, 60
lock-free data structure, 63
locking strategy, 466
lock table, 95
loosely coupled components, 451

M
many-core accelerators, 410
many-core architectures, 30
many-cores scalability, 380
Map, 139
MapOverlap, 139
mappers, 413
mapping, 36, 52
mapping strategy, 435
MapReduce, 139, 246, 411
Map skeleton, 122
massively parallel applications, 29
master, 115
master/worker pattern, 192, 432
memory allocator, 69
memory bandwidth, 431, 466
memory hierarchy, 466
memory latency, 345–346, 355
memory reclamation, 70
memory wall, 10
Mercurium, 109
Message Passing Interface (MPI), 37, 102, 229, 432
message passing libraries, 7
message-passing paradigm, 466
metadata storage, 219–220
microkernels, 453
miscompression rate, 392
MPI. See Message Passing Interface (MPI)
MPI applications, 432
MPI communication, 116
multicore architectures, 9, 101
multicore clusters, 432
multicore nodes, 431
multiobjective evolutionary algorithms, 303
multiobjective optimization, 302
Multiple Instruction Multiple Data, 6
Multiple Instruction Single Data, 6
multiprocessors, 5
multiprogramming, 5
multithreaded applications, 366
multithreading, 8
mutation operator, 304
mutual exclusion, 60, 81

N
Nanos++, 109
nested transaction, 91
NUMA, 355

O
object-orientation, 185
off-chip communication, 407
offline profiling, 211
offload function, 418
offloading, 253
off-the-shelf components, 408
OmpSs programming model, 46, 102
on-chip memory, 391
opacity, 86
OpenCL, 47, 129
OpenCL/CUDA, 102
OpenMP, 37, 102, 212, 230
OPL, 46
optimistic concurrency control, 83
optimizations, 36

P
parallel bug patterns, 325
parallel design patterns, 191, 265
parallel implementation, 344
parallelism, 3
parallel performance, 343
parallel statements, 189
partitioned scheduling, 463
performance, 32
performance aware, 249
performance bottlenecks, 96
performance metrics, 432, 467
performance optimization, 344
performance portability, 243
performance predictions, 243
pessimistic concurrency control, 83
PetaBricks, 246
pipeline parallelism, 186
pipelining, 4, 124
polymorphism, 177
population, 305
portability, 32, 121
POSIX, 293, 467
power wall, 10
predictability, 367
prefetching, 4
priority queue, 67
productivity, 32
programmability, 143
programmability gap, 32
programming models, 6, 32

Q
queue, 66

R
read sharing, 171
read-write ratio, 466
real-time applications, 467
real-time scheduling, 463, 465
Reduce, 139
reducers, 413
Reduction, 123
region tree, 113
regression analysis, 381
repeatability, 367
resource aware, 249
resource configurations, 415
review techniques, 324
runtime system, 102

S
scalability, 228, 363, 367, 432
scalability tests, 366
scalar processor, 4
scheduling, 36, 52
semantics, 84
sensitivity analysis, 395
sequential implementation, 344
Sequoia, 45
serializability, 85
set, 68
shared cache, 431
shared memory, 8, 303
shared-memory communication, 466
shared-memory locking, 466
shared-memory multicore processor, 343
shared memory multiprocessor, 6
shared-memory paradigm, 466
shared-memory parallel programming, 343
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>shared-memory programming</td>
<td>231</td>
</tr>
<tr>
<td>Simics</td>
<td>393</td>
</tr>
<tr>
<td>single global lock</td>
<td>85</td>
</tr>
<tr>
<td>Single Instruction Single Data</td>
<td>5</td>
</tr>
<tr>
<td>single-system image</td>
<td>455</td>
</tr>
<tr>
<td>skeletal approach</td>
<td>265</td>
</tr>
<tr>
<td>skeleton</td>
<td>104</td>
</tr>
<tr>
<td>skeleton programming</td>
<td>122</td>
</tr>
<tr>
<td>skip list</td>
<td>68</td>
</tr>
<tr>
<td>SLICES</td>
<td>161</td>
</tr>
<tr>
<td>socket</td>
<td>348</td>
</tr>
<tr>
<td>software accelerators</td>
<td>266</td>
</tr>
<tr>
<td>software pipelining</td>
<td>222</td>
</tr>
<tr>
<td>software transactional memory</td>
<td>81</td>
</tr>
<tr>
<td>SP@CE</td>
<td>44</td>
</tr>
<tr>
<td>SpecFP2000</td>
<td>393</td>
</tr>
<tr>
<td>speculative computation</td>
<td>209</td>
</tr>
<tr>
<td>speculative parallelization</td>
<td>206</td>
</tr>
<tr>
<td>speculative thread</td>
<td>207</td>
</tr>
<tr>
<td>speedup</td>
<td>310, 381, 432, 435</td>
</tr>
<tr>
<td>splitting</td>
<td>191</td>
</tr>
<tr>
<td>SPMD applications</td>
<td>433</td>
</tr>
<tr>
<td>stack</td>
<td>65</td>
</tr>
<tr>
<td>STAPL</td>
<td>246</td>
</tr>
<tr>
<td>Star Superscalar</td>
<td>232</td>
</tr>
<tr>
<td>state separation</td>
<td>208</td>
</tr>
<tr>
<td>stream graph</td>
<td>186</td>
</tr>
<tr>
<td>StreamIt</td>
<td>144, 201</td>
</tr>
<tr>
<td>stream parallelism</td>
<td>263</td>
</tr>
<tr>
<td>stream programming</td>
<td>143, 185</td>
</tr>
<tr>
<td>superscalar processor</td>
<td>4</td>
</tr>
<tr>
<td>symmetric multiprocessors</td>
<td>7</td>
</tr>
<tr>
<td>synchronization</td>
<td>60, 61, 114, 185, 339, 344, 350, 465</td>
</tr>
<tr>
<td>synchronization APIs</td>
<td>358</td>
</tr>
<tr>
<td>synchronization overheads</td>
<td>358</td>
</tr>
<tr>
<td>synchronization primitives</td>
<td>353</td>
</tr>
<tr>
<td>synchronous data flow</td>
<td>143</td>
</tr>
<tr>
<td>system performance</td>
<td>364</td>
</tr>
<tr>
<td><strong>T</strong></td>
<td></td>
</tr>
<tr>
<td>task dependency graph</td>
<td>113</td>
</tr>
<tr>
<td>task farm</td>
<td>124</td>
</tr>
<tr>
<td>task graph</td>
<td>116</td>
</tr>
<tr>
<td>task parallelism</td>
<td>186</td>
</tr>
<tr>
<td>test environment</td>
<td>378</td>
</tr>
<tr>
<td>testing</td>
<td>323</td>
</tr>
<tr>
<td>thread-based programming</td>
<td>229</td>
</tr>
<tr>
<td>Threading Building Blocks</td>
<td>138</td>
</tr>
<tr>
<td>thread-level parallelism</td>
<td>205</td>
</tr>
<tr>
<td>threads</td>
<td>37, 345</td>
</tr>
<tr>
<td>throughput</td>
<td>380, 465</td>
</tr>
<tr>
<td>timing requirements</td>
<td>465</td>
</tr>
<tr>
<td>trace</td>
<td>106</td>
</tr>
<tr>
<td>transaction</td>
<td>166</td>
</tr>
<tr>
<td>transactional abort</td>
<td>82</td>
</tr>
<tr>
<td>transactional boosting</td>
<td>177</td>
</tr>
<tr>
<td>transactional commit</td>
<td>82</td>
</tr>
<tr>
<td>transactional isolation</td>
<td>87</td>
</tr>
<tr>
<td>transactional memory</td>
<td>119, 165</td>
</tr>
<tr>
<td>transactional polymorphism</td>
<td>178</td>
</tr>
<tr>
<td>transaction models</td>
<td>177</td>
</tr>
<tr>
<td>transaction nesting</td>
<td>169</td>
</tr>
<tr>
<td>transformation</td>
<td>153</td>
</tr>
<tr>
<td>tree</td>
<td>69</td>
</tr>
<tr>
<td>try-to-eat procedure</td>
<td>167</td>
</tr>
<tr>
<td>tuning</td>
<td>185</td>
</tr>
<tr>
<td>two-phase locking</td>
<td>171</td>
</tr>
<tr>
<td><strong>U</strong></td>
<td></td>
</tr>
<tr>
<td>unified runtime architecture</td>
<td>119</td>
</tr>
<tr>
<td>unit of transfer</td>
<td>390</td>
</tr>
<tr>
<td><strong>V</strong></td>
<td></td>
</tr>
<tr>
<td>video encoding</td>
<td>281</td>
</tr>
<tr>
<td><strong>W</strong></td>
<td></td>
</tr>
<tr>
<td>waiting mechanism</td>
<td>351</td>
</tr>
<tr>
<td>WebLogic</td>
<td>363</td>
</tr>
<tr>
<td>work block</td>
<td>189</td>
</tr>
<tr>
<td>work descriptors</td>
<td>112</td>
</tr>
<tr>
<td>worksharings</td>
<td>106</td>
</tr>
<tr>
<td>work unit</td>
<td>418</td>
</tr>
<tr>
<td>write-back policy</td>
<td>115</td>
</tr>
<tr>
<td><strong>X</strong></td>
<td></td>
</tr>
<tr>
<td>XJava</td>
<td>187</td>
</tr>
<tr>
<td>XtreemOS</td>
<td>460</td>
</tr>
</tbody>
</table>