INDEX

**Numbers**

2D grids, 53–60
2D threads, 53–57
2D wave equation, stencil calculation, 400–401

**A**

access, shared memory, 208–211
accessMatrix.cu, 151–152
achieved occupancy, 100
active blocks, 89
active warps, 89
nvprof and, 100
algebra operations, cuSPARSE, 333
aligned coalesced memory access, 159
aligned memory access, 158–160
allocating memory, 146
shared memory, 206
ALU (arithmetic logic unit), 71–72
AoS (array of structures), 171–176
APIs (application program interfaces)
  driver API, 14–15
  runtime API, 14–15
APOD development cycle, 426–429
assessment stage, 427
deployment stage, 429
optimization stage, 428
parallelization stage, 427–428
applications
  compute-bound, 300
  I/O-bound, 300
  legacy, 311
  scaling, GPU clusters, 409–421
architecture
  compute capability, 77–78
heterogeneous
  device code, 9
  hardware accelerator, 10
  host code, 9
parallel computing and, 2
arithmetic logic unit (ALU), 71–72
arrays
  summation, 28–29
values, exchange across warps, 261–262
assessment stage (APOD development cycle), 427
asynchronous behavior of functions, 268
asynchronous control functions, OpenACC, 381–382
asynchronous streams, 275
atomic floating-point values, 320–321
atomic instructions, 304–306, 315–321
  built-in, 317–318
  performance, 318–321
axpyi function, 333

**B**

bandwidth, 7, 10
effective, 179
global memory access, 239
latency hiding and, 91
matrix transpose, 180–194
diagonal transpose, 189–192
lower bound, 181–185
naive transpose, 185–187
thin blocks and, 193–194
unroll transpose, 187–189
upper bound, 181–185
memory bandwidth, 179–180
nvprof, 440
performance, 176–179
theoretical, 179–180
BLAS (Basic Linear Algebra Subprograms), 341.
    See also cuBLAS library
block partitioning, 5
blockIdx, 31
blocking stream, 275–276
blocks. See also thread blocks
branch divergence
    interleaved pairs and, 112–114
    loop unrolling and, 115–120
    parallel reduction, 106–111
    template functions and, 120–122
warp and, 117–118
branch efficiency, 86
breadth-first scheduling, overlapping kernels and, 293–294
BSR (block compressed sparse row), 337
BSRX (Extended BSR), 337
built-in atomic instructions, 317–318
butterfly exchange, warps, 260

C programs, porting to CUDA, 462–463
crypt, 463–475
cache memory, 142
    GPU versus CPU, 213
    program-managed cache, shared memory, 205
cacheConfig argument, 213
cached memory load, 161–163
    read-only cache, 168–169
callback functions, stream callbacks, 295–297
CAS (compare-and-swap) operator, 315–316
checkDeviceInfor.cu, 61–62
checkGlobalVariable( ) function, 143
checking memory, nvprof, 100–101
checkInnerArray( ) function, 183
checkInnerStruct( ) function, 179
checkResult( ) function, 39
checkThreadIndex.cu, 51–53
child grid, 123–124
child thread, 123–124
child thread block, 123–124
clauses, loop directive, 372–374
clusters, 7
coalesced memory access, 158–160, 239–240
    baseline transpose kernel, 240–241
    parallelism, 249
shared memory
    matrix transpose, 241–249
    matrix transpose and, 241–249
course-grain concurrency, 269
column-major order
    rectangular shared memory, 226–227
    square shared memory, 218–220
    reading, 220–221
compiler directives, OpenACC, 366
compiling
    flags, instruction generation, 312–313
    separate, 433
    whole-program, 433
compute capability, 11
    architecture specifications, 77–78
    resource limits, 89
compute directives, OpenACC
    kernels, 367–370
    loop, 372–374
    parallel, 370–371
computer architecture, 2
    bandwidth, 7
    clusters, 7
    gfflops, 7
    latex, 7
    many-core, 7
MIMD (Multiple Instruction Multiple Data), 6
MISD (Multiple Instruction Single Data), 6
multicore, 7
SIMD (Single Instruction Multiple Data), 6
SIMT (Single Instruction, Multiple Thread), 8
SISD (Single Instruction Single Data), 6
throughput, 7
counter-bound applications, 300
concurrency
coarse-grain, 269
concurrency-limiting resources, GPUs, 284–286
grid level, 267
kernel level, 267
non-NULL streams, 279–281
configuration
events, synchronous streams, 278
execution configuration, 36
shared memory access mode, 212
shared memory amount, 212–213
constant memory, 137, 140–141, 250
further reading, 490–491
read-only cache, global memory access and, 253–255
stencils, implementing, 250–253
COO sparse matrix, 337
COO (coordinate) sparse matrix, 334
core, 3
CPU (central processing unit), 3
versus GPU, 8
host code, 25
timer, kernels, 44–47
CPU thread versus GPU thread, 12
cpuSecond function, 44
crypt
assessing, 463–464
deploying, 472
multi-GPU, 473
OpenMP-CUDA, 473–475
optimizing, 465–472
parallelizing, 464–465
CSC (compressed sparse column), 337
CSR (compressed sparse row) format, 334–337
cuBLAS library, 341–342
data management, 342–343
development, 345–346
example, 343–345
versus MLK BLAS, 362–363
CUDA C
ANSI C and, 14
kernels, 15
programming difficulty, 20–21
CUDA cores, 71–72
SMs, 71–72
CUDA Occupancy Calculator, 95
CUDA Toolkit, CUDA Occupancy Calculator, 95
cudaDeviceCanAccessPeer() function, 391
cudaDeviceEnablePeerAccess() function, 392
cudaDeviceGetSharedMemConfig() function, 212
cudaDeviceGetStreamMemConfig() function, 273
cudaDeviceProp, 61, 94
cudaDeviceProp structure, 61
cudaDeviceReset() function, 19
cudaDeviceSetCacheConfig() function, 212
cudaDeviceSynchronize, 97
cudaDeviceSynchronize() function, 37, 44, 276
cudaErrorInvalidDevicePointer.cudaFree(
) function, 146
cudaErrorMemoryAllocation, 146
cudaEventCreateWithFlags() function, 278
cudaEventQuery() function, 274
cudaEventRecord() function, 274
cudaEventSynchronize() function, 274
cudaFree() function, 26, 146
cudaFreeHost() function, 150, 155
cudaFuncSetCacheConfig() function, 140, 213
cuda-gdb, kernel debugging and, 448–449
cudaGetDeviceCount() function, 389
cudaGetDeviceProperties, 94
cudaGetDeviceProperties() function, 61, 94
cudaGetLastError() function, 27, 438
cudaGetLastError() function, 437
cudaGetSymbolAddress() function, 145
cudaHostAlloc() function, 155, 156
cudaHostGetDevicePointer() function, 156
cudaMalloc function, 26–27, 29
cudaMallocHost() function, 149, 155–156
cudaMallocManaged() function, 163
cudaMemcpy() function, 146
cudaMemcpy function, 27, 30, 145
cudaMemcpyDeviceToHost() function, 147
cudaMemcpyHostToDevice, 147
cudaMemcpyToSymbol() function, 141, 250
cudaMemset() function, 26

cudaPeekLastError( ) function, 438

cudaSetDevice( ) function, 390

cudaStreamAddCallback( ) function, 296

cudaStreamCreate( ) function, 269–270

cudaStreamCreateWithFlags( ) function, 276

cudaStreamCreateWithPriority( ) function, 273

cudaStreamDestroy( ) function, 270

cudaStreamWaitEvent( ) function, 278

CUDA_VISIBLE_DEVICES variable, 206

cuFFT library, 346–349

cuFFT API, 347–348

equivalent, 348–349

versus FFTW, 363–364

versus MKL, 363–364

cufft.cu, 348

cufftXtMalloc( ) function, 360

cufftXtMemcpy( ) function, 360

cuRAND library

APIs, 351–354

development, 357–358

equivalent, 354–357

pseudo-random numbers, 349–350

quasi-random numbers, 349–350

curandCreateGenerator( ) function, 351

curandSetPseudoRandomGeneratorSeed( ) function, 352

curandSetQuasiRandomGeneratorDimensions ( ) function, 353

cuSPARSE library, 332–341

algorithms, 333

data storage formats, 333–337

BSR (block compressed sparse row), 337

BSRX (Extended BSR), 337

COO sparse matrix, 334, 337

CSC (compressed sparse column), 337

CSR (compressed sparse row), 337

CSR (compressed sparse row) format, 334–336

dense matrix, 334, 337

ELL (Ellpack-Itpack), 337

HYB (hybrid), 337

development, 340

formatting conversion, 337–338

versus MKL (Math Kernel Library), 361–362

cusparse.cu, 338–339

CWD (CUDA Work Distributor), 293

cyclic partitioning, 5, 428

data dependency, 3

data directive, 375–379

data directives, OpenACC, data, 375–379

data parallelism, 4

data partitions, 5

data race, 304–305

data sharing, warps, 258–262

data storage, cuSPARSE library, 333–337

BSR (block compressed sparse row), 337

BSRX (Extended BSR), 337

COO sparse matrix, 334, 337

CSC (compressed sparse column), 337

CSR (compressed sparse row), 337

CSR (compressed sparse row) format, 334–336

dense matrix, 334, 337

ELL (Ellpack-Itpack), 337

HYB (hybrid), 337

data transfer

CPU-to-CPU, 410

CPU affinity, 412–413

inter-node MPI communication, 411–412

cudaMemcpy function, 27

GPU-to-GPU

CUDA-aware MPI, 416–417

traditional MPI, 413–416

overlapping kernels

breadth-first scheduling, 293–294

depth-first scheduling, 289–293

deallocating memory, 146

debugging

cuda-memcheck

compiling for, 456–457

error types, 457–458

kernels

assert, 455–456

cuda-gb, 452–454

cuda-gdb, 448–449

D

data dependency, 3

data directive, 375–379

data directives, OpenACC, data, 375–379
environment information, 450–451
focus, 449–450
memory inspection, 450
printf, 454–455
tunable parameters, 451–452
racecheck, 458–462
declaration
events, 273–274
memory for multi-GPUs, 393–394
dense matrix, 334, 337
dependency
false dependencies, 272, 281
Fermi GPUs, 281–283
inter-stream dependencies, 288–289
deployment stage (APOD development cycle), 429
depth-first scheduling, overlapping kernels and, 289–293
development environment, 21
device
data movement, 28
memory transfer to/from host, 150
parallelism, 92
device code, 9, 25
device management, 60
functions, OpenACC, 381
GPU
determining best, 63
nvidia-smi for queries, 63–64
runtime API, 61–63
setting at runtime, 64–65
device memory, 143
functions, 26
kernels, 26
_device_ qualifier, 38
diagonal transpose, 189–192
directives. See compiler directives
dispatching operations, OpenMP and, 283–284
divergence, branch divergence
interleaved pairs and, 112–114
parallel reduction, 106–111
template functions and, 120–122
domains, library support, 329–330
dotci function, 333
doti function, 333
double-precision floating-point values, 306–308
DRAM (dynamic random access memory), 137
constant memory, 250
driver API, 14–15
Drop-In Libraries, 358–359
Dynamic Parallelism, 76, 122–123
Hello World, 124–128
nested execution, 123–124
nested reduction, 128–132
restrictions, 128
dynamic shared memory, 221–222
parallel reduction of global access, 238–239
dynamically declared memory
padding, 223–225
rectangular, 228–229
effective bandwidth, 179
global memory access, 239
eligible warps, 89
ELL (Ellpack-Itpack), 337
environment variables
CUDA_VISIBLE_DEVICES, 206
streams and, 284–286
error handling, 40, 437–438
events
configuration, synchronous streams, 278
declaring, 273–274
elapsed time, 274–275
recording, 274–275
execution
configuration, 36
further reading, 489–490
multi-GPUs, 389–391, 406–409
overlap, 294–295
explicit synchronization of streams, 277–278
false dependencies, 272, 284
Fermi GPUs, 281–283
_fdivef function, 311
Fermi architecture, 71–73
false dependencies, 281–283
FFT (fast Fourier transform), 346. See also cuFFT library

FFTW (Fastest Fourier Transformation in the West), 363–364
files, integration to C project, 436–437
finite difference, multi-GPUs, 400–409
flags, instruction generation, 312–313
floating-point MAD (FMAD), 311–312
floating-point unit (FPU), 71–72
floating-point values
atomic, 320–321
double-precision, 306–308
instructions, 301–303
representable, 302
single-precision, 306–308
floating-point-perf.cu, 323
FMAD (floating-point MAD), 311–312
_fmul, 313–314
FPU (floating-point unit), 71–72
functions
asynchronous behavior, 268
axpyi, 333
checkGlobalVariable( ), 143
checkInnerArray( ), 183
checkInnerStruct( ), 179
checkResult( ), 39
cpuSecond, 44
cudaDeviceCanAccessPeer( ), 391
cudaDeviceEnablePeerAccess( ), 392
cudaDeviceGetSharedMemConfig( ), 212
cudaDeviceGetStreamPriorityRange( ), 273
cudaDeviceReset( ), 19
cudaDeviceSetCacheConfig( ), 212
cudaDeviceSynchronize( ), 37, 44, 276
cudaErrorInvalidDevicePointer.
cudaFree( ), 146
cudaEventCreateWithFlags( ), 278
cudaEventQuery( ), 274
cudaEventRecord( ), 274
cudaEventSynchronize( ), 274
cudaFree( ), 26, 146
cudaFreeHost( ), 150, 155
cudaFuncSetCacheConfig( ), 140, 213
cudaGetDeviceCount( ), 389
cudaGetDeviceProperties( ), 61, 94
cudaGetErrorString( ), 27, 438
cudaGetLastError( ), 437
cudaGetSymbolAddress( ), 145
cudaHostAlloc( ), 155, 156
cudaHostGetDevicePointer( ), 156
cudaMalloc, 26–27, 29
cudaMallocHost( ), 149, 155–156
cudaMallocManaged( ), 163
cudaMemcpy, 27, 30, 145
cudaMemcpy( ), 146
cudaMemcpyDeviceToHost( ), 147
cudaMemcpyToSymbol( ), 141, 250
cudaMemset( ), 26
cudaPeekLastError( ), 438
cudaSetDevice( ), 390
cudaStreamAddCallback( ), 296
cudaStreamCreate( ), 269–270
cudaStreamCreateWithFlags( ), 276
cudaStreamCreateWithPriority( ), 273
cudaStreamDestroy( ), 270
cudaStreamWaitEvent( ), 278
cufftXtMalloc( ), 360
cufftXtMemcpy( ), 360
curandCreateGenerator( ), 351
curandSetPseudoRandomGeneratorSeed( ), 352
curandSetQuasiRandomGenerator
Dimensions( ), 353
device memory, 26
dotci, 333
doti, 333
_foodivdef, 311
gtimeofday( ), 44
gthr, 333
gthrz, 333
initialInnerArray( ), 182
initialInnerStruct( ), 179
intrinsic, 303–304
visualizing, 309–311
kernel functions, 37–39
type qualifiers, 38
mm, 333
mv, 333
nextafterf( ), 302
printf, 34
printThreadInfo, 51
rand(), 354–357
roti, 345
scstr, 345
setColReadCol(), 219–220
setRowReadColDyn(), 222
sm, 333
standard, 303–304
visualizing, 309–311
sumArraysOnGPU(), 39
sumMatrixOnHost(), 53–54
sv, 333
synchronous behavior, 27, 268
_syncthreads(), 140
template, 120–121
type qualifiers, 38

gang parallelism (OpenACC), 365
GeForce (NVIDIA), 10
getimeofday() system call, 44
gflops, 7
GigaThread engine, 72
__global__ declaration, 37–38
global memory, 137, 141–142
further reading, 490
nvprof, 441
parallel reduction
dynamic shared memory and, 238–239
shared memory and, 232–236
unrolling, 236–238
read-only cache access, 253–255
static global memory, 143–145
global memory read, 160–169
global memory write, 169–171
__global__ qualifier, 38
globalVariable.cu, 144
GMU (Grid Management Unit), 293
GPU, 1
versus CPU, 8
data size, 11–12
determining best, 63
device code, 9, 25
device management, 60
determining best, 63
nvidia-smi for queries, 63–64
runtime API, 61–63
setting at runtime, 64–65
Fermi, 11
hardware accelerator, 10
Hello World, 17–19
Kepler, 11
many-core architecture, 8
nvidia-smi, 63–64
parallelism level, 11–12
querying, 63–64
GPU architecture
concurrency-limiting resources, 286–287
Fermi architecture, 71–73
Kepler architecture, 73–78
profiling, 78–80
SM (Streaming Multiprocessors), 68
GPU thread versus CPU thread, 12
GPUDirect, 419–421
GPU-to-GPU data transfer, 410
CUDA-aware MPI, 416–417
GPU affinity, 413–414
GPUDirect RDMA, 419–421
intra-node MPI communication, 417–418
intranode with CUDA-aware MPI, 417–418
traditional MPI, 413–415
grid level concurrency, 267, 268
streams and, 268
 grids
 2D, 53–57
CWD (CUDA Work Distributor), 293
dimensions, 32–36
 matrices, 53–60
 threads, 31
  matrix indexing, 49–53
__gthr function, 333
__gtzr function, 333

halo region, 389
handling errors, 40, 437–438
hardware accelerator, 10
Harvard architecture, 2
Hello World, 17–19
nested, 124–128
heterogeneous computing, 8–9
architecture
  device code, 9
  host code, 9
data size, 11–12
further reading, 489
hardware accelerator, 12
parallelism level, 11–12
hierarchy
  memory hierarchy, 27
    Kepler GPUs, 204
    principle of locality, 136
threads, 36
host, 25
data movement, 28
memory functions, 26
memory transfer to/from device, 150
host code, 9
  _host_summation, 28–29
  __host__ qualifier, 38
host threads, multi-GPUs and, 394
HPC (high-performance computing), 1
HYB (hybrid), 337
Hyper-Q, 76
  stream scheduling, 272

I
if...then, 82–83
ILP (instruction-level parallelism), 92
  implementation
    development, APOD, 426–429
    further reading, 492
implicit synchronization of streams, 276–277
indexing, matrices, 49–53
InfiniBand network, 388
initialInnerArray() function, 182
initialInnerStruct() function, 179
in-place storage, 104
instruction dispatch units, 72–73
instructions, 300–301
  atomic, 304–306, 315–321
build-in, 317–318
  performance, 318–321
double-precision floating-point, 306–308
floating-point values, 301–303
generating, manipulating generation, 311–315
optimization, 306–324
parallelism and, 432
single-precision floating point, 306–308
interleaved pairs, branched divergence and, 112–114
inter-stream dependencies, 288–289
intrinsic functions, 303–304
  visualizing, 309–311
I/O-bound applications, 300

K
Kepler architecture, 73–78
  Hyper-Q, 272
  memory hierarchy, 204
  kernel functions, 37–39
    type qualifiers, 38
  kernel level concurrency, 267
kernels, 15
  bandwidth, memory bandwidth, 179–180
  concurrent execution, non-NULL streams, 279–281
debugging
  assert, 455–456
cuda-gb, 452–454
cuda-gdb, 448–449
  environment information, 450–451
  focus, 449–450
  memory inspection, 450
  printf, 454–455
tunable parameters, 451–452
device memory, 26, 440
  execution configuration, 36
L1/L2 cache, 440
  launching, 36–37
matrix addition, 151–152
overlapping
  breadth-first scheduling, 293–294
  depth-first scheduling, 289–293
reduceInterleaved, 115
shared memory, 440
system memory, 440
texture cache, 440
timing, 43
   CPU timer, 44–47
   nvprof, 47–49
verification, 39
writing, 37–39
kernels directive, 367–370

L
 latency, 7
   bandwidth and, 91
   maximizing throughput, 90
   minimizing, 90
   throughput and, 91
   latency hiding, 90–93
   launching, kernels, 36–37
legacy applications, 311
libraries, 327
   advantages, 329
   configuration, 331–332
   cuBLAS, 341–342
   cuFFT, 346–349
   cuRAND, 349–358
   cuSPARSE, 332–341
device memory
   allocating, 331
   populating with inputs, 331
   results retrieval, 332
domains supported, 329–330
Drop-In Libraries, 358–359
further reading, 491
handles, 331
inputs, converting to supported format, 331
multi-GPU, 359–361
native format, 332
OpenACC, 382–384
performance survey, 361–364
resources, releasing, 332
workflow, 330–332
LLVM infrastructure, 16
local memory, 137, 139
locality, 20
spatial, 136
temporal, 136
loop directive, 372–374
loop unrolling, 114–120
   branch divergence and, 115–117
lower bound of transpose kernels, 181–185

M
 MAD (multiply-add), 300
 FMAD (floating-point MAD), 311–312
managed memory, 157
   performance comparisons, 196–197
many-core, 7
matrices
   indexing, 49–53
   summing
      1D grids and blocks, 57–58
      2D grids and 1D blocks, 58–60
      2D grids and blocks, 53–57
   transpose, coalesced memory access, 241–249
matrix addition kernel, 151–152
   partial matrix addition, 152–155
   thin blocks, 193–194
   unified memory and, 195–199
matrix transpose, 180–194
   diagonal transpose, 189–192
   lower bound, 181–185
   naive transpose, 185–187
   unroll transpose, 187–189
   upper bound, 181–185
memory
   bank conflicts, 441–442
   checking, nvprof, 100–101
   constant memory, 137, 250
      stencils, 250–253
   data race, 304–305
debugging
   cuda-memcheck, 456–458
   racecheck, 458–462
device memory, 143
   functions, 26
   kernels, 26
DRAM (dynamic random access memory), 137
dynamic, parallel reduction, 238–239
global memory, 137
hierarchy, 27
  Kepler GPUs, 204
  principle of locality, 136
loads, 161
  cached load, 161–163
  uncached load, 163–164
local memory, 137
managed memory, 157
multi-GPUs, declaring, 393–394
pageable, 148
page-locked, 148–149
read-only cache, 168–169
registers, 137
shared, 20
  access mode, 208–211
  access mode configuration, 212
  coalesced memory access, 241–249
  configuring amount, 212–213
  dynamic, 221–222
  padding, 211–212
  parallel reduction, 232–238
  rectangular, 225–232
SMs, 70–71
  square, 217–225
  synchronization, 214–216
shared memory, 137, 139–140, 206, 216–217
  allocation, 206
  banks, 206
  on-board memory, 204
  on-chip memory, 204
  partitions, 205
  program-managed cache, 205
  SMs and, 70–71
  thread blocks, 217
uses, 204
SRAM (static random access memory), 137
static global memory, 143–145
texture memory, 137
Unified Memory, 157–158
unsafe access, 304–305
virtual, 148
memory access patterns, 158
  aligned access, 158–160
  AoS (array of structures), 171–176
  coalesced access, 158–160
global memory read, 160–169
global memory write, 169–171
performance tuning, 176–178
SoA (structure of arrays), 171–176
memory allocation, `cudaMalloc` function, 26
memory bandwidth, 10, 179–180
memory fence, 215–216
memory management, 145–146
  allocation, 146
  deallocation, 146
functions, OpenACC, 382
matrix addition kernel, 151–152
  partial matrix addition, 152–155
pinned memory, 148–150
  zero copy memory, 150–155
transfer, 146–148
  to/from device, 150
  to/from host, 150
Unified Memory, 157–158
UVA (Unified Virtual Addressing), 156–157
memory model
  memory variables, 142–143
  non-programmable memory, 137
  cache memory, 142
programmable memory, 137
  constant memory, 137, 140–141
  global memory, 137, 141–142
  local memory, 137, 139
  registers, 137, 138–139
  shared memory, 137, 139–140
  texture memory, 137, 141
memory spaces, 30
memory variables, 142–143
`memTransfer.cu`, 147–148
messages, scaling applications, 418–419
  `--metrics` flag, 86
MIMD (Multiple Instruction Multiple Data), 6
MISD (Multiple Instruction Single Data), 6
MKL (Math Kernel Library), 361–362
  `versus` cuFFT, 363–364
MLK BLAS, `versus` cuBLAS, 362–363
mm function, 333
MPI (Message Passing Interface), 409–410
  GPU-to-GPU data transfer, 413–416
multicore, 3, 7
multi-GPU, 387
   2D wave equation, stencil calculation, 400–401
   compiling, 395–396, 406–409
   converting to, 388–389
   finite difference, 400–409
   further reading, 491–492
   memory, declaring for multiple devices, 393–394
   overlapping and, 405–406
   P2P
      enabling access, 391–392, 396
      memory copy, 392, 396–397
   patterns, 401–403
   synchronization, 392–393
   threads, distribution from single host, 394
   Multi-GPU Libraries, 359–361
   multiply-add. See MAD (multiply-add)
   mv function, 333
   MVAPICH2, 410

N
naive transpose, 185–187
nested execution, 123–124
   Hello World, 124–128
nested reduction, 128–132
   nestedHelloWorld.cu, 125–127
   nestedReduce.cu, 129–130
   nestedReduceNoSync.cu, 130–131
   nextafterf( ) function, 302
non-blocking streams, 276
non-NULL streams, 269
   asynchronous behavior, 275
   coarse-grain concurrency, 269
   concurrent kernels, 279–281
   synchronous behavior, 275
non-programmable memory, 137
   cache memory, 142
   Nsight Eclipse Edition, 79
   NULL streams, 269
   nvcc compiler, 15, 18
      LLVM infrastructure, 16
   NVIDIA, tools extension, 446–448

NVIDIA GPU
   compute capability, 11
   GeForce, 10
   nvcc compiler, 15
   Quadro, 10
   Tegra, 10
   Tesla, 10
   nvidia-smi, 63–65
   nvprof, 47–49, 79
   active warps and, 100
   checking memory, 100–101
   event/metric summary mode, 439
   event/metric trace mode, 439
   global memory access, 441
   instruction throughput, 443
   memory bandwidth, 440
   memory bank conflicts, 441–442
   profiling bank conflicts, 439–440
   profiling scope, 440
   register spilling, 442
   summary mode, 439
   trace mode, 439
   nvvp, 443–445

O
occupancy, 93–96
   achieved occupancy, 100
   on-board memory, 204
   on-chip memory, 204
   OpenACC, 328
      combining with libraries, 382–384
      compiler directives, 366
      compute directives
         kernels, 367–370
         loop, 372–374
         parallel, 370–371
   data directives, data, 375–379
   gang parallelism, 365
   gang-partitioned mode, 366
   gang-redundant mode, 365
   gangs, 365
   platform model, 365
   runtime API, 380

491
asynchronous control functions, 381–382
device management functions, 381
memory management functions, 382
runtime initialization functions, 382
vector parallelism, 365
vector-partitioned mode, 366
vector-single mode, 366
worker parallelism, 365
worker-partitioned mode, 366
workers, 365
vector elements, 365
vector width, 365
worker-single mode, 366
OpenMP, 20, 24
dispatched operations, 283–284
operations
dispatched, OpenMP and, 283–284
kernel launches, 275
memory-related, 275
optimization, 429–430
instruction execution, 432
memory access, 430–432
*nnvp*, 443–445
parallelism, 430
profile-driven, 78–80, 438–439
*nnprof*, 439–443
optimization stage (APOD development cycle), 428
organizing threads, 30–36
overlapping execution, 294–295
overlapping kernels
breadth-first scheduling, 293–294
depth-first scheduling, 289–293
overlapping multi-GPUs, 405–406
dynamically declared, 223–225, 229–230
statically declared, 223, 229–230
pageable memory, 148
page-locked memory, 148–149
parallel computing, 2
locality, 20
parallel directive, 370–371
parallel programming, 2
parallel programs, 3–4
parallel reduction, 104–106
branch divergence, 106–111
dynamic shared memory and, 238–239
shared memory and, 232–236
unrolling and, 236–238
warp shuffle instruction, 262–264
parallel threads, 49
synchronization
explicit barrier, 214–215
memory fence, 215–216
volatile qualifier, 216
weakly-ordered memory model, 214
parallelism. See also Dynamic Parallelism
coalesced memory access, 249
data parallelism, 4
device, 92
gang parallelism (OpenACC), 365
grid/block heuristics, 98–99
optimization and, 430
performance and, 176–177
SM, arithmetic utilization, 92
sufficient, 93
task parallelism, 4
vector parallelism (OpenACC), 365
worker parallelism, 365
parallelization stage (APOD development cycle), 427–428
parallelizing, *crypt*, 464–465
parent grid, 123–124
parent thread blocks, 123–124
parent threads, 123–124
partial matrix addition kernel, 152–155
*partialsMatr.cu*, 154
partitioning

P2P, multi-GPUs
enabling access, 391–392, 396
memory copy, 392, 396–397
UVA and, 398–400
padding memory, 211–212
shared
block partitioning, 5, 428

cyclic partitioning, 5, 428

resources, 87–89

shared memory, 205

patterns, multi-GPUs, 401–403

PCIe node, 388–389

peak computational performance, 10

performance

atomic instructions, 318–321

libraries, 361–364

memory bandwidth, 176–179

rectangular shared memory, 231–232

square shared memory, 225

pinMemTransfer.cu, 150

pinned memory, 148–150

zero copy memory, 150–155

pluggability, 328

porting C programs to CUDA, 462–463

crypt

assessing, 463–464

deploying, 472–475

optimizing, 465–472

parallelizing, 464–465

principle of locality, 136

printf function, 34

printThreadInfo function, 51

priorities, streams, 273

profile-driven optimization, nvprof, 443–445

profiling, 78–80

program structure, 19

programmable memory, 137

constant memory, 137, 140–141

global memory, 137, 141–142

local memory, 137, 139

registers, 137, 138–139

shared memory, 137, 139–140

texture memory, 137, 141

program-managed cache, 205

programming difficulty, 20–21

programming model, 23–24

further reading, 489

programming structure, 25–26

pseudo-random numbers, 349–350

pthread, 20

ptreads, 24

PTX (Parallel Thread Execution), 309

Quadro (NVIDIA), 10

quasi-random numbers, 349–350

racecheck, 458–462

rand() function, 354–357

random numbers, cuRAND library, 349–350

RDMA (Remote Direct Memory Access), 419–421

read memory with offset, 165–168

readOffsetUnroll4() kernel, 187

read-only cache, 168–169

global memory access, 253–255

readSegment.c, 170–172

readSegmentUnroll.c, 185

recording, events, 274–275

rectangular shared memory, 225–232

column-major order access, 226–227

column-major order reading, 227–228

dynamically declared, 228–229

performance, 231–232

row-major order access, 226–227

row-major order writing, 227–228

reduceInteger.cu, 107–110

reduceInterleaved, 115

reduceInterleaved kernel, 115

reduceNeighboredLess, 112

reduction, parallel reduction, 104–106

register spilling, 139, 442

registers, 137, 138–139

resources

compute capability and, 89

concurrency-limiting, 286–287

partitioning, 87–89

RNG algorithms, 350

roti function, 333

row-major order
rectangular memory, 227–228
rectangular shared memory, 226–227
shared memory access, 218–220
shared memory writing, 220–221
runtime API, 14–15
GUP device management, 61–63
OpenACC
asynchronous control functions, 381–382
device management functions, 381
memory management functions, 382
runtime initialization functions, 382
runtime initialization functions, OpenACC, 382

S

scalability, warps, 98
scaling applications
CPU-to-CPU data transfer, 410
  CPU affinity, 412–413
  inter-node MPI communication, 411–412
GPU-to-GPU data transfer
  CUDA-aware MPI, 416–417
  GPUDirect RDMA, 419–421
  intranode with CUDA-aware MPI, 417–418
  traditional MPI, 413–416
MPI (Message Passing Interface), 409–410
schedule, Hyper-Q, 272
scheduling streams, 271
false dependencies, 272
`scot` function, 333
selected warps, 89
separate compilation, 433, 434
  Makefile, 434–436
sequential programs, 3–4
`setColReadCol( )` function, 219–220
`setRowReadColDyn( )` function, 222
SFUs (special function units), 72
shared memory, 20, 137, 139–140, 204–205
  access mode, 208–211
  configuration, 212
  allocation, 206
banks, 206
  conflict, 207–208
coalesced memory access, matrix transpose and, 241–249
configuring amount, 212–213
data layout, 216–225
dynamic, 221–222
further reading, 490–491
on-board memory, 204
on-chip memory, 204
padding, 211–212, 229–230
parallel reduction of global access, 232–236
partitions, 205
program-managed cache, 205
rectangular, 225–232
column-major order access, 226–227
column-major order reading, 227–228
dynamically declared, 228–229
performance, 231–232
row-major order access, 226–227
row-major order writing, 227–228
SMs and, 70–71
square
column-major order access, 218–220
column-major order reading, 220–221
declaring, 217
dynamic memory, 221–222
dynamically declared, padding, 223–225
performance, 225
row-major order access, 218–220
row-major order writing, 220–221
statically declared, padding, 223
synchronization
  explicit barrier, 214–215
  memory fence, 215–216
  volatile qualifier, 216
  weakly-ordered memory model, 214
thread blocks, 205
uses, 204
SIMD (Single Instruction Multiple Data), 6, 68–69
simpleDeviceQuery.cu, 94–95
simpleDivergence.cu, 84–85
simpleMathAoS_Aligned.cu, 184
simpleMathAoS.cu, 173–176
SIMT (Single Instruction, Multiple Thread), 8, 68
warp, 80
single-precision floating-point values, 306–308
SISD (Single Instruction Single Data), 6
SM (Streaming Multiprocessors), 68
CUDA cores, 71–72
parallelism, arithmetic utilization, 92
shared memory, 70–71
thread blocks, 70
sm function, 333
SISD (Single Instruction, Single Data), 6
SoA (structure of arrays), 171–176
spatial locality, 20, 136
square shared memory
column-major order access, 218–220
column-major order reading, 220–221
dynamic memory, 221–222
dynamically declared, padding, 223–225
performance, 225
row-major order access, 218–220
row-major order writing, 220–221
statically declared, padding, 223
SRAM (static random access memory), 137
stalled warps, 89
standard functions, 303–304
visualizing, 309–311
static global memory, 143–145
statically declared memory, padding, 223
stencils
2D wave equation, multi-GPU and, 400–401, 403–405
constant memory, 250–253
stream callbacks, 295–297
stream priorities, 273
streams, 268–269
asynchronous, 275
blocking streams, 275–276
default, blocking behaviors, 287–288
dispatching operations, OpenMP and, 283–284
environment variables, 284–286
events, 273–274
false dependencies, 284
Fermi GPUs, 281–283
grid level concurrency and, 268
inter-stream dependencies, 288–289
non-blocking, 276
non-NULL, 269
coarse-grain concurrency, 269
NULL, 269
scheduling, 271
false dependencies, 272
Hyper-Q, 272
synchronization, 275–276
blocking streams, 275–276
event configuration, 278
explicit, 277–278
implicit, 276–277
non-blocking streams, 275–276
synchronous, 275
strerror( ) function, 27
suggested reading, 489–492
sumArrayOnGPU-small-case.cu, 40–42
sumArraysOnGPU( ) function, 39
sumArraysOnGPU-timer.cu, 45–46
sumArrayZeroCopy.cu, 151–154, 156
sumArrayZeroCopyUV.cu, 162
sumMatrix, 100–101
sumMatrix.cu, 99
sumMatrixGPU.cu, 446
sumMatrixGPUManaged.cu, 195, 205–206
sumMatrixOnGPU-1D-grid-1D-block.cu, 58
sumMatrixOnGPU2D.cu, 98
sumMatrixOnGPU2D kernel, 59
sumMatrixOnGPU-2D-grid-2D-block.cu, 54–56
sumMatrixOnHost( ) function, 53–54
sv function, 333
switch-case structure, 121
synchronization, 275–276
multi-GPUs, 392–393
warp, 96–98
synchronous behavior of functions, 268
synchronous streams, 275
blocking streams, 275–276
events, configuration, 278
explicit, 277–278
implicit, 276–277
nonblocking streams, 275–276
_syncthreads, 97, 106–107
_syncthreads() function, 140

T

_ syncthreads – Visual Profiler

task parallelism, 4
tasks, 3
Tegra (NVIDIA), 10
template function, 120–121
temporal locality, 20, 136
Tesla (NVIDIA), 10
compute capabilities, 11
texture memory, 137, 141
theoretical bandwidth, 179–180
thin blocks, 193–194
thread blocks, 31
2D, 53–57
active blocks, 89
dimensions, 32–36
hardware view, 82
logical view, 82
matrices, 53–60
matrix indexing, 49–53
shared memory, 205
SMs and, 70
warps and, 80–82
threadIdx.x, 31, 81–82
threads
grids, 31
matrix indexing, 49–53
hierarchy, 36
host, multi-GPUs and, 394
OpenMP, 20, 24
organizing, 30–36
parallel, 49
pthread, 20, 24
throughput, 7
latency hiding and, 91
time, events, 274–275
timing, kernels, 43
CPU timer, 44–47
nvprof, 47–49
TLP (thread-level parallelism), 92
tools, 446–448
transferring memory, 146–148
to/from device, 150
to/from host, 150
transparent scalability, 98
transpose.cu, 182–184
types, kernel functions, 38

U

uncached memory load, 163–164
Unified Memory, 25, 157–158
unified memory, matrix addition and, 195–199
uniprocessor, 3
unroll transpose, 187–189
unrolling
parallel reduction of global access, 236–238
performance and, 176–177
unsafe access to memory, 304–305
upper bound of transpose kernels, 181–185
UVA (Unified Virtual Addressing), 156–157, 387
P2P memory access, 398–400

V

variables
environment, streams and, 284–286
environment variables, CUDA_VISIBLE_DEVICES, 206
memory variables, 142–143
vector parallelism (OpenACC), 365
vectors, summation, 40–43
verification, kernels, 39
virtual memory, 148
Visual Profiler, 79, 443–444
Guided analysis, 444
Unguided analysis, 444–445
volatile qualifier, 216

warps
active warps, 89

nvprof and, 100
array values exchange, 260–261
branch divergence and, 117–118
butterfly exchange, 260
data sharing, 258–262
divergence, 82–87
eligible, 89
lane index, 255
lanes, 255
latency hiding, 90–93
occupancy, 93–96
resource partitioning, 87–89
scalability, 98

X-Y-Z

-Xcompiler, 29

schedulers, 72–73
selected warps, 89
shift down, 259
shift up, 258–259
shift within, 259–260
shuffle instruction, 255
parallel reduction, 262–264
variants, 256–257
stalled, 89
synchronization, 96–98
thread blocks and, 80–82
values, broadcasting, 258
weakly-ordered memory model, 214
whole-program compilation, 433
worker parallelism (OpenACC), 365
writeSegment.cu, 176–177