Index

Numbers
32-bit applications, file handling in, 364–367
32-bit architecture, specifying in compiler, 103
32-bit code on UltraSPARC processors, 30
64-bit architecture, specifying in compiler, 103
286 processor, 39
386 processor, 39
SSE instruction sets on, 95–96
-xtarget=generic compiler option, 99
80286 processor, 39
80386 processor, 39
SSE instruction sets on, 95–96
-xtarget=generic compiler option, 99
8086 processor, 39

A
.a adornment (SPARC instruction), 25
.a files (static libraries), 182
access to memory
  error detection for, 261–262, 274
  metrics on (example), 288–290
accessing matrix data, 347–348
active_cycle_count counter, 310
activity reports. See system status reporting
addition operations, 9
address, memory, 16, 18
address space for process, reporting on, 78–79
addressing modes, SPARC, 24–25
aggressively optimized code, compiler
options for, 93–95. See also Sun Studio compiler, using
algorithmic complexity, 437–442
alias_level pragma, 143–144
alias pragma, 144–145
aliasing control pragmas, 142–147
aliasing of pointers. See pointer aliasing in C
  and C++
align pragma, 136–137
-aligncommon=16 compiler option, 101
alignment of variables, specifying,
  135, 136–137
AMD64 processors, 40
  compiler performance, 105
Amdahl's Law, 377, 440, 443
analyzer GUI, 207, 210–212
AND operation (logical), 9
application profiles. See profiling
  performance
applications
  calls from, reporting on, 80–81
  compiler options for. See Sun Studio
  compiler, using
debbuging, 265–271
  running under dbx, 268–271
disassembling, 89
looking for process IDs for, 61–62
multithreaded. See multithreading
processes as, 371. See also processes
reporting on, 84–92
disassembles, examining, 89
file contents, type of, 86–87
library linkage, 84–86
library version information, 87–89
metadata in files, 90–92
segment size, reporting, 90
symbols in files, 87
runtime linking of, information on,
  191–192
segments in, reporting size of, 90
selecting target machine type for,
  103–107
tuning, 442–444. See also optimization;
profiling performance
ar tool, 184
architecture, specifying with compiler,
  99, 104, 105, 106–107
archives (static libraries), creating, 184
arguments passed to process, reporting on,
  79
arrays, VIS instructions for searching,
  203–205
assembly language (machine code), 18–19.
  See also ISA (information set
  architecture)
assertion flags, 443
associativity, cache, 12–13
atomic directive, 432–433
atomic operations, 395–396
ATS (Automatic Tuning and
  Troubleshooting System), 271–274
attributed time for routines, 214
audit interface, linker, 192
automatic parallelization, 408–409, 424–425
Automatic Tuning and Troubleshooting
  System (ATS), 271–274
autoscooping, 404–405
ax_miss_count_dm counter, 310
ax_miss_count_dm_if counter, 310
ax_read_count_dm counter, 310
B
  -B compiler option, 183
  bandwidth, memory, 14–15
    source code optimizations, 326–327
    synthetic metrics for, 292–293
  bandwidth, system bus, 15, 308
  base pointer (x86 code), 43, 44, 264
bcheck command, 256–257
big-endian vs. little-endian systems, 41–42
binding, controlling processor use with,
  52–53
BIT tool, 224–226
bounds checking, 256–257
  runtime array bounds (Fortran), 259
Br_completed counter, 308
Br_taken counter, 308
branch delay slot, 25
branch instructions, 25
  events, performance counters for,
    300–301, 315–316
branch pipes, 7, 9–10
  UltraSPARC III processors, 31
branches
  if statements, 357–364
    loop unrolling and, 320–321
    mispredicted branches, 10, 107
    performance counters for,
      300–301, 315–316
    predictors, 10, 358
breakpoints (dbx), setting, 269
bsdmalloc library, 193–194
buses, 15–16
  reporting activity of, 76
busstat command, 76–77, 308
bypass (storing data to memory), 299, 352
byte ordering, x64 processors, 41–42
C
  C- and C++-specific compiler optimizations,
    123–135
  -xalias_level compiler option,
    127–133, 409
-xalias_level=basic option, 129–130, 426
-C compiler option, 259
C99 floating-point exceptions, 169–170
cache configuration, setting, 104, 105
cache latency (memory latency), 8–9, 34–35
measuring, 288–290, 333–334
source code optimizations, 333–337
fetching integer data, 328
synthetic metrics for, 290–292
caches, 4, 11–14
access metrics (example), 288–290
clock speed and, 5
data cache events, 283–285
data cache lines, showing, 229
instruction cache events, 283–285
L3 cache, 302–303
link time optimization, 115
miss events, cycles lost to, 287
prefetch cache, 293–295
prefetch for cache lines, 335–337
thrashing in, 12–13, 349–353
UltraSPARC III and IV processors, 31–33
call instructions, SPARC, 25
call stack. See stack
caller–callee information, 212–214
tail-call optimization and, 236–237
calls from applications, reporting on, 80–81
cc command (f90 command), 99
character I/O, reporting on, 67
checking code. See code checking and debugging
child processes, 378–379
chip multiprocessing (CMP), 373
chip multithreading (CMT), 6, 373. See also multithreading
atomic operations, 395–396
mutexes with, 394
optimizing for, 446
parallelization, 377
code checking and debugging,
244–245, 247–276, 274–276
ATS for locating optimization bugs, 271–274
compile-time checking, 248–255
compiler options for code checking
C language, 248, 250
C++ language, 250, 251–252
compiler options for debugging, 93–95
dbx debugger, 262–271
commands mapped to mdb, 276
compiler options for debugging, 262
debug information format, 263–264
debugging application (example), 265–268
running applications under, 268–271
running on core files, 264–265
including debug information when compiling, 102–103
mdb debugger, 274–276
multithreaded code, 413–416
runtime checking, 256–262
code complexity, 437–442
code coverage information
dtrace command, 241–244
tcov command, 239–241
to validate profile feedback, 114–115
code layout optimizations, 107–116
code parallelization. See parallelization
code scheduling, 105
specifying, 8–9, 104, 106
coding rules, 437
collect command, 207
collect utility, 207, 208–210, 243, 280
hardware performance counters, 218
collision rate, 70
command-line performance analysis. See er_print tool
commandline arguments, examining, 79
common subexpression elimination (CSE), 325
communication costs of parallelization, 377.
See also parallelization
compare operations, 9
comparisons in floating-point operations, 158
compile-time array bounds checking, 259
compile-time code checking, 248–255
compile time, optimization level and, 94, 96
compiler commentary, 244–245
compiling for Performance Analyzer, 210
detecting data races, 412
complex instruction set computing. See CISC processor
complexity, algorithmic, 437–442
complexity, parallelism, 444
compressed structure code, 341–342
conditional breakpoints (dbx), 269
conditional move statements, 358–360
conditional statements, 10, 357–364. See also branches
configuration of system, reporting on, 49–55
conflict (memory), multiple streams and, 348
constants, floating-point in single precision, 172–173
contended locks, 395–396
count flag, Opteron performance counters, 311
count Perl script, 280–281
current processes. See processes
cycle budget (UltraSPARC T1), 305–307
Cycle_cnt counter, 282
cycle_count counter, 310
cycles (clock speed), 4, 5
lost of processor stall events, 299
lost to cache miss events. See misses, cache
stalled, 6, 316–317
RAW recycles, 352–354
store queue, 354–357
UltraSPARC III processors, 34

D
-D_FILE_OFFSET_BITS compiler option, 366
-D_LARGEFILE_SOURCE compiler option, 366
-dalign compiler option, 122–123
data cache. See also caches
memory bandwidth measurements, 292–293
memory latency measurements
example of, 288–290
synthetic metrics for, 290–292
reorganizing data structures, 339–343
showing data cache lines, 229
data density, 15
data prefetch. See prefetch instructions
data races in multithreaded applications, 389–392, 412–413, 434
data reads after writes, 352–354
data space profiling, 226–233
data streams
multiple, optimizing, 348–349
storing, 329
data structures, optimizing, 339–349
algorithmic complexity, 437–442
matrices and accesses, 347–348
multiple streams, 348–349
prefetching, 343–346
reorganizing, 339–343
various considerations, 346
Index

data TLB misses, 351–352

dbx debugger, 262–271
  commands mapped to mdb, 276
  compiler options for debugging, 262
  debug information format, 263–264
  debugging application (example), 265–268
  running applications under, 268–271
  running on core files, 264–265

DC_access counter, 313
DC_dtib_L1_miss_L2 counters, 314
DC_miss counter, 304, 309, 313
  multipliers for conversion to cycles, 306

DC_rd counter, 283
DC_rd_miss counter, 283, 290
DC_refill_from_L2 counter, 312
DC_refill_from_system counter, 312
DC_wr counter, 283
DC_wr_miss counter, 283

delays (floating-point), hoisting, 163–165
divides (integer), avoiding, 174–177

division by zero, 11
  handler for (example), 168–169
  zero divided by zero. See NaN (Not-a-Number)
division operations, 9
does_not_read_global_data pragma, 137–138
  errno variable and, 166
does_not_write_global_data pragma, 137–138
  errno variable and, 166
doors, 380
double-precision floating-point registers, SPARC, 24, 29–30
double-precision values, 150–151
  not promoting single precision to, 171–172
dprofiling, 226–233

dumping, 71–72
die, CPU, 4
direct mapped caches, 12
directories (files), reporting disk space used by, 70
directory mechanism, 16
  UltraSPARC III processors, 36
dis command, 89
  disabling virtual processors, 51
disassembles, examining, 89
discover tool, 261–262
disk drive reporting, 57
  space utilization, 71–72

Dispatch0_2nd_br counter, 300
Dispatch0_br_target counter, 300
Dispatch0_IC_miss counter, 285
Dispatch0_mispred counter, 300
dives (floating-point), hoisting, 163–165
divides (integer), avoiding, 174–177
disassembly, 25
dependence analysis, 120–121, 141
detailed warnings enabled, 252
developer tools. See also specific tool by name
code checking. See code checking and debugging
floating-point optimization, 149–180. See also floating-point calculations
  compiler options for, 149–173
  integer maths, 174–178
  Kahan Summation Formula, 161–163
  multiply accumulate instructions, 173–174
  parameter passing with SPARC V8, 178–180
reorganizing data structures, 341–342

informational tools, 49–92. See also
task-specific tools
application reporting, 84–92
  system configuration reporting, 49–55
  on system status. See system status reporting
libraries and linking. See libraries and linking
performance profiling tools. See profiling performance

Sun Studio compiler. See Sun Studio compiler, using
dwarfdump command, 90–92
dump command, 72
dumpstabs command, 90–91, 264
dtrace command, 233–234, 241–244
dtrex command, 233–234

Dynamic Debugging Tools (DBT), 261–265
Dynamic Debugging Tools (DBT) command, 262–271
Dynamic Debugging Tools (DBT) commands mapped to mdb, 276
Dynamic Debugging Tools (DBT) compiler options for debugging, 262
Dynamic Debugging Tools (DBT) debug information format, 263–264
Dynamic Debugging Tools (DBT) debugging application (example), 265–268
Dynamic Debugging Tools (DBT) running applications under, 268–271
Dynamic Debugging Tools (DBT) running on core files, 264–265
Dynamic Debugging Tools (DBT) DC_access counter, 313
Dynamic Debugging Tools (DBT) DC_dtib_L1_miss_L2 counters, 314
Dynamic Debugging Tools (DBT) DC_miss counter, 304, 309, 313
  multipliers for conversion to cycles, 306
Dynamic Debugging Tools (DBT) DC_rd counter, 283
Dynamic Debugging Tools (DBT) DC_rd_miss counter, 283, 290
Dynamic Debugging Tools (DBT) DC_refill_from_L2 counter, 312
Dynamic Debugging Tools (DBT) DC_refill_from_system counter, 312
Dynamic Debugging Tools (DBT) DC_wr counter, 283
Dynamic Debugging Tools (DBT) DC_wr_miss counter, 283
development tools. See also specific tool by name
code checking. See code checking and debugging

Floating-Point Optimization (FPO), 149–180. See also floating-point calculations
  compiler options for, 149–173
  integer maths, 174–178
  Kahan Summation Formula, 161–163
  multiply accumulate instructions, 173–174
  parameter passing with SPARC V8, 178–180
reorganizing data structures, 341–342

Informational Tools (IT), 49–92. See also

libraries and linking

Profiling tools. See profiling performance

Sun Studio compiler. See Sun Studio compiler, using

df command, 71–72
die, CPU, 4
direct mapped caches, 12
directories (files), reporting disk space used by, 70
directory mechanism, 16
  UltraSPARC III processors, 36
dis command, 89
  disabling virtual processors, 51
disassembles, examining, 89
discover tool, 261–262
disk drive reporting, 57
  space utilization, 71–72

Dispatch0_2nd_br counter, 300
Dispatch0_br_target counter, 300
Dispatch0_IC_miss counter, 285
Dispatch0_mispred counter, 300
dives (floating-point), hoisting, 163–165
divides (integer), avoiding, 174–177
division by zero, 11
  handler for (example), 168–169
  zero divided by zero. See NaN (Not-a-Number)
division operations, 9

does_not_read_global_data pragma, 137–138
  errno variable and, 166
does_not_write_global_data pragma, 137–138
  errno variable and, 166
doors, 380
double-precision floating-point registers, SPARC, 24, 29–30
double-precision values, 150–151
  not promoting single precision to, 171–172
dprofiling, 226–233

Dynamic Debugging Tools (DBT) command, 262–271
Dynamic Debugging Tools (DBT) commands mapped to mdb, 276
Dynamic Debugging Tools (DBT) compiler options for debugging, 262
Dynamic Debugging Tools (DBT) debug information format, 263–264
Dynamic Debugging Tools (DBT) debugging application (example), 265–268
Dynamic Debugging Tools (DBT) running applications under, 268–271
Dynamic Debugging Tools (DBT) running on core files, 264–265
Dynamic Debugging Tools (DBT) DC_access counter, 313
Dynamic Debugging Tools (DBT) DC_dtib_L1_miss_L2 counters, 314
Dynamic Debugging Tools (DBT) DC_miss counter, 304, 309, 313
  multipliers for conversion to cycles, 306
Dynamic Debugging Tools (DBT) DC_rd counter, 283
Dynamic Debugging Tools (DBT) DC_rd_miss counter, 283, 290
Dynamic Debugging Tools (DBT) DC_refill_from_L2 counter, 312
Dynamic Debugging Tools (DBT) DC_refill_from_system counter, 312
Dynamic Debugging Tools (DBT) DC_wr counter, 283
Dynamic Debugging Tools (DBT) DC_wr_miss counter, 283
development tools. See also specific tool by name
code checking. See code checking and debugging

Floating-Point Optimization (FPO), 149–180. See also floating-point calculations
  compiler options for, 149–173
  integer maths, 174–178
  Kahan Summation Formula, 161–163
  multiply accumulate instructions, 173–174
  parameter passing with SPARC V8, 178–180
reorganizing data structures, 341–342

Informational Tools (IT), 49–92. See also

libraries and linking

Profiling tools. See profiling performance

Sun Studio compiler. See Sun Studio compiler, using

0138134553_Gove.book  Page 451  Tuesday, December 4, 2007  2:29 PM
E
-E compiler option, 250
EC_ic_miss counter, 285
EC_misses counter, 286
EC_rd_miss counter, 286, 290
EC_ref counter, 286
elfdump command, 90–92
EMT64 processors, 40
compiler performance, 105
emulated floating-point instructions, reporting number of, 64
diendianess, 42
equality of NaN as untrue, 156
.er directories, 208
er_kernel tool, 233–235
er_print tool, 207, 214–215
er_src utility, 244–245, 405
errno variable, 165–166
errors, checking for. See code checking and debugging
exceptions from floating-point calculations, 167–170
exclusive time in routines, 211
execution frequency statistics, 240–241
execution latency, 8
execution pipes, 7–11
named, 380
UltraSPARC III processors, 30–31
experiment repository, 208
exponent (floating-point numbers), 153–154
f
f90 command (cc command), 99
FA_pipe_completion counter, 282
false sharing, 397–399
-fast compiler option, 98–102
errno variable and, 165
optimizations in, 100–102
mathematical optimizations, 178–180
specifying target architecture, 99, 105
FGA and FGM pipes (SPARC), 31
file access system routines, reporting on, 67
file command, 86–87
file contents
metadata, reporting on, 90–92
reporting type of, 86–87
symbols defined in, reporting, 87
file descriptor limits, 364–366
file directories. See directories
file-level scope, specifying alias level, 144
filling registers, 28, 40
fill traps, 77
loop splitting and, 321–322
finalization code in libraries, 187
::find_leaks command (dbx), 276
findbug script (ATS), 273–274
fini pragma, 187
first level (L1) caches, UltraSPARC III, 31, 34
floating_instructions counter, 310
floating-point calculations. See also floating-point optimization
aggressive code optimizations and, 94
-fast compiler optimizations, 101
assessing operations with performance counters, 281–283
catching exceptions with dbx, 270
counting instructions executed, 304
exception flags, 167–170
executing time, reporting on UltraSPARC processors, 217
hoisting of divides, 163–165
honoring parentheses in, 158–159
IEEE-754 standard, 11, 150–151
inline template versions, 170–171
optimization of, 157–158
parentheses, honoring, 165
reductions and, 406
reordering, 159–161
Kahan Summation Formula, 161–163
specifying which events cause traps, 166–167
subnormal numbers, 153–155
flushing to zero, 155
unnecessary, avoiding, 158–159
vectorizing, 151–153
floating-point constants, in single precision, 172–173
floating-point instructions, emulated, 64
floating-point optimization, 149–180. See also floating-point calculations
compiler options for, 149–180
Kahan Summation Formula, 161–163
multiply accumulate instructions, 173–174
Index

parameter passing with SPARC V8, 178–180
reorganizing data structures, 341–342
floating-point pipes, 7, 11
UltraSPARC III processors, 31
flush directive, 432–433
flushing subnormal numbers to zero, 155
FM_pipe_completion counter, 282
fma function (C99), 174
-fma=fused compiler option, 174
-fnstore compiler option, 150
-fns compiler option, 155
within -fast option, 101, 150
fork() function, 378–382
Fortran compiler optimizations, 135–136
controlling amount of warning information, 253
Fortran runtime array bounds checking, 259
FP_instr_cnt counter, 304
multipliers for conversion to cycles, 306
fpversion command, 55, 105
FR_dispatch_stall_performance counters, 316–317
FR_dispatch_stalls counter, 316
FR_nothing_to_dispatch counter, 316
FR_reired_performance counters, 315–316
frame pointer (x86 code), 43, 44, 264
free memory, reporting on, 56, 67
free operations, 194
-fsimple compiler option, 157–161, 163–165
within -fast option, 101, 150
-fsingle compiler option, 171–172
within -fast option, 101, 150
ftime function, 199
-trap compiler option, 166–167
within -fast option, 101, 150
-trap=common option, 101, 150, 166
-trap=%none option, 101, 150, 166
funcs command, 271
functions
cloning, 325–326
infrequently called, specifying as, 138–139
performance of. See profiling performance side effects of, 137–138
specifying non-access to global data, 137–138
timing duration of, 199–201
functions (func) command, 214–215
fused multiply accumulate instructions, 173–174
G
-g and -g0 compiler options, 102, 210, 262, 263
compiler commentary, 244–245
-G compiler option, 184
GCC compatibility, 147
general compiler optimizations, 116–123
in -fast compiler option, 101
general-purpose registers, x86 processors, 44–45
general system information, reporting on, 49–51
generic target for compilation, 104–105
get_tick function, 199–200
gethrtime function, 199, 200
getimeofday function, 199
global affects of functions, 138–139
global data, specifying function's access to, 137–138
global registers, SPARC, 23, 26–27
global variables
  link time optimization, 115
  pointers and, 125–126
gprof command, 237–239
  granularity, mutex, 394
H
-H compiler option, 250
hardware
  characteristics of SPARC processors, 55
  performance counters, 73–76, 208, 304–305
values read from. See volatile variables
hardware prefetch, 32, 294–295, 297
header files inclusion information, 250
heap, specifying page size for, 123
hoisting loop-invariant code, 324–325
hoisting of floating-point divides, 163–165
horizontal scaling, 375–376
hot regions of code, identifying, 443–444
I

I/O activity, reporting, 68–69
  character I/O, 67
IA-32 instruction set, 40, 42–43
IC_fetch counter, 312
IC_instr_fetch_stall counter, 316
IC_itib_L1_miss_L2_counters, 314
IC_miss_cancelled counter, 285
IC_miss_counter, 285, 304, 309, 312
  multipliers for conversion to cycles, 306
IC_ref counter, 285
IC_refill_from_L2 counter, 312
IC_refill_from_system counter, 312
ICXs (involuntary context switches),
  reporting on, 60, 63
identifying processes, 58
idle time, reporting, 57
IEEE-754 standard, 11, 150–151
ieee_flags function, 167
ieee_handler function, 168
if_r_iu_req_mi_go counter, 310
if statements, 357–364
  conditional move statements, 358–360
  misaligned memory accesses (SPARC),
    361–364
impdep2_instructions counter, 310
in-order execution processors, 5
inclusive time in routines, 211
indexing, 18
information set architecture. See ISA
informational tools, 49–92
application reporting, 84–92
process- and processor-specific reporting
  bus activity, 76
  commandline arguments, examining, 79
  files held open by processes, 79
  hardware performance counters,
    73–76, 208, 304–305
  stack dumps, 79–80
  timing process execution, 72–73
  tracing application execution, 80–81
  trap activity, 77–78
  user and system code activity, 82–84
  virtual memory mapping, 78–79
system configuration reporting, 49–55
on system status. See system status
  reporting
infrequently used code, moving, 112
init pragma, 187
initialization code in libraries, 187
inline template versions of floating-point
  functions, 170
inlining across source files. See crossfile
  optimization
inlining routines, 108
  copying or moving memory, 338–339
  profile feedback for, 112
input registers, SPARC, 23, 24, 26–27
  register windows and, 27–28
Instr_cnt counter, 282, 309
Instr_FGU_arithmetic counter, 308
Instr_ld counter, 309
Instr_st counter, 309
instruction cache
  events, performance counters for,
  layout of, 107
  unrolled loops and, 321
instruction_counts counter, 310
instruction set, specifying in compiler,
  99, 104, 105, 106–107
instruction set extensions, x64 processors,
  40, 46
instruction set reports, 53
instructions (processors)
  execution speed of, 4–5
  latency of execution, 8
integer maths
  with floating-point values, 174–178
  prefetch and, 327–328
integer operation pipes, 7, 9
integer pipes, UltraSPARC III processors, 31
integer registers, SPARC, 26–27
integer registers, x86, 44
interchanging loops, 322–324
interleaving, 35
interposition of libraries, 189–191
interprocess cross calls, 63
interrupts, reporting number of, 57
Intimate Shared Memory (ISM), 380
invert flag, Opteron performance counters, 311
involuntary context switches (ICXs), 60, 63
iostat command, 68–69
ISA (information set architecture), 18–19
libraries for specific, searching for, 186
SPARC processors, 19, 23–30
supported, outputting, 53–55
x64 architecture, 42–44
extensions to, 40, 46
isalist command, 53, 86
$ISALIST symbol, 186
ISM (Intimate Shared Memory), 380
ITLB_miss counter, 304, 309
IU_Stat_Br_count_taken counter, 300
IU_Stat_Br_count_untaken counter, 300
IU_Stat_Br_miss_taken counter, 300
IU_Stat_Br_miss_untaken counter, 300

K
Kahan Summation Formula, 161–163
kernel
exploring system code activity, 82–84
memory allocation, reporting on, 67
statistics on, reporting, 64–68
traps. See traps
-Kpic and -KPIC compiler options, 185
kstat command, 64

L
-l compiler option, 183
-L compiler option, 183, 185–186
L1 caches, 31, 34
L2 caches
fetching integer data, 327–328
memory bandwidth measurements, 292–293
memory latency measurements
example of, 288–290
synthetic metrics for, 290–292
performance counters, 286–287, 304
UltraSPARC II processors, 34
UltraSPARC III and IV+ processors, 31–32, 33, 34
L2_dmiss_ld counter, 304, 309
multipliers for conversion to cycles, 306
L2_imiss counter, 304, 309
multipliers for conversion to cycles, 306
L3 cache, 302–303
la_objopen function, 192
la_version function, 192
large files, in 32-bit applications, 366–367
latency, memory, 8–9, 34–35
measuring, 288–290, 333–334
source code optimizations, 333–337
fetching integer data, 328
synthetic metrics for, 290–292
latency groups (l-groups), 36
latency of instruction execution, 8
lazy loading of libraries, 187
-lbsmalloc compiler option, 193
-lbtmalloc compiler option, 193
LD_AUDIT environment variable, 193
LD_DEBUG environment variable, 191–192
LD_LIBRARY_PATH environment variables, 85, 186
LD_LIBRARY_PATH_32 variable, 190–191
LD_LIBRARY_PATH_64 variable, 190–191
LD_LIBRARY_PATH_32 variable, 186
LD_LIBRARY_PATH_64 variable, 186
ld linker, 182
LD_OPTIONS environment variable, 191–192
LD_PRELOAD environment variable, 85, 190–191, 257–258
LD_PRELOAD_32 variable, 190–191
LD_PRELOAD_64 variable, 190–191
ldd command, 84–86
libc library, 183, 193
libc_per library, 193
libcpc interface, 280
libfast library, 193–194, 196
libmopt library, 171
libmvect library, 152
libraries, specific, 193–198
bsdmalloc library, 193–194
libc library, 183, 193
libc_per library, 193
libfast library, 193–194, 196
libmopt library, 171
libmvect library, 152
libumem library, 193, 194, 258–259, 274
MediaLib library, 202
memory management libraries, 193–196
libraries, specific (Continued)
  mtmalloc library, 194, 195, 259
  performance library, 196–197
  Rogue Wave and STLport4 libraries, 198
  Sun Math Library, 201–202
  watchmalloc.so library, 257–258
libraries and linking, 87, 181–205. See also
  libraries, specific
    audit interface, 192–193
    debug interface, 191–192
    disassembling libraries, 89
    dynamic and static libraries, 184–185
    dynamic and static linking, 182–183
    how to link libraries, 183
    initialization and finalization code, 187
    lazy loading of libraries, 185–187
    libraries linked to applications, reporting, 84–86
    library calls, 199–205
      searching arrays with VIS instructions, 203–205
      SIMD instructions and MediaLib, 202
      for timing, 199–201
      using most appropriate, 201–202
    library interposition, 189–191
    link-time optimization, 108, 115–116
    locations of libraries, specifying, 185–187
    overview of linking, 181–182
    recognizing standard library functions (compiler), 133–135
    segments in, reporting size of, 90
    symbol scoping, 188
    versions of libraries, 87–89, 186
    -library=stlport4 compiler option, 198
    libumem library, 193, 194, 258–259, 274
    lightweight processes (LWPs), 59, 76
    limit command, 214–215
    limit stacksize command, 136
    line sizes, caches, 12, 13
    link-time optimization, 108, 115–116
    linking. See libraries and linking
    lint command, 248–250
    little-endian vs. big-endian systems, 41–42
    -lm9x compiler option, 169
    load balancing/imbalance, 52
    OpenMP specification and, 429–430
    load operations, counting, 126
    load_store_instructions counter, 310
load/store pipe, 8–9
  UltraSPARC III processors, 31
loads (processors), 9
  physical and virtual addresses of, 229
  SPARC operations, 24
local registers, SPARC, 23, 26–27
  register windows and, 27–28
local variables
  aligning for optimal layout, 135, 136–137
  placing on stack, 135–136
  locking objects to multiple changes. See
    mutexes
logical operations, 9, 358–360
  loops
    compiler commentary on, 244–245
    dependence analysis. See dependence
      analysis
    fusion, 321–322
    hoisting of floating-point divides, 163–165
    interchange and tiling, 322–324
    invariant hoisting, 324–325
    parallelizing (OpenMP), 403–406
    splitting, 322
    unrolling and pipelining, 140–142, 320–321
  -lsunmath compiler option, 201
  -lunem compiler option, 193
  LWPs (lightweight processes), 59, 76
M
  -M compiler option, 110
  -m32 compiler option, 106
  -m64 compiler option, 99, 106
  machine code (assembly language), 18–19.
    See also ISA (information set
      architecture)
  macro options, 98
  makefiles, dependency information for, 251
  malloc command requests, counting, 81–83
  MALLOC_DEBUG environment variable, 258
  malloc operations, 193–196
    debugging options under, 258–259
  mantissa (floating-point numbers), 153–154
  manual performance optimizations,
    avoiding, 441–442
<table>
<thead>
<tr>
<th>Terms</th>
<th>Page Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapfiles, generating</td>
<td>108, 110–111</td>
</tr>
<tr>
<td>master threads (See Pthreads)</td>
<td></td>
</tr>
<tr>
<td>matrices, optimizing</td>
<td>347–348</td>
</tr>
<tr>
<td>may_not_point_to pragma</td>
<td>146–147</td>
</tr>
<tr>
<td>may_point_to pragma</td>
<td>145–146</td>
</tr>
<tr>
<td>MC_reads_0_sh counter</td>
<td>303</td>
</tr>
<tr>
<td>mcs command, generating</td>
<td>92</td>
</tr>
<tr>
<td>mdb debugger, commands mapped to dbx</td>
<td>274–276</td>
</tr>
<tr>
<td>MediaLib library</td>
<td>202</td>
</tr>
<tr>
<td>Megahertz Myth</td>
<td>5</td>
</tr>
<tr>
<td>MEMBAR instructions</td>
<td>16, 36</td>
</tr>
<tr>
<td>memcpy command</td>
<td>338</td>
</tr>
<tr>
<td>memory access error detection, addressing</td>
<td>261–262, 274</td>
</tr>
<tr>
<td>bandwidth, source code optimizations, synthetic metrics for</td>
<td>14–15, 292–293</td>
</tr>
<tr>
<td>cache latency, measuring</td>
<td>8–9, 34–35</td>
</tr>
<tr>
<td>source code optimizations, synthetic metrics for</td>
<td>328, 333–337</td>
</tr>
<tr>
<td>misaligned accesses (SPARC), ordering (x64 processors)</td>
<td>121–123, 361–364</td>
</tr>
<tr>
<td>copying and moving</td>
<td>338–339</td>
</tr>
<tr>
<td>dependencies on, specifying none</td>
<td>141</td>
</tr>
<tr>
<td>misaligned accesses (SPARC), changing what applications request,</td>
<td>54</td>
</tr>
<tr>
<td>errno variable and, setting page size, supported page size,</td>
<td>123, 53–55</td>
</tr>
<tr>
<td>thrashing in caches, page size and, 350–351</td>
<td></td>
</tr>
<tr>
<td>tagging, threaded applications, virtual</td>
<td>385, 399–402</td>
</tr>
<tr>
<td>virtual</td>
<td>16–18</td>
</tr>
<tr>
<td>mapping information, reporting for process, utilization of,</td>
<td>78–79, 56–57</td>
</tr>
<tr>
<td>memory footprint, compiling for 32- or 64-bit architecture,</td>
<td>103</td>
</tr>
<tr>
<td>memory management libraries, message queues, metadata in files,</td>
<td>223</td>
</tr>
<tr>
<td>metadata on performance reports, memory bandwidth measurements,</td>
<td>292–293</td>
</tr>
<tr>
<td>memory latency measurements, example of, synthetic metrics for,</td>
<td>288–290</td>
</tr>
<tr>
<td>TLB (Translation Lookaside Buffer), MMX instruction set extensions,</td>
<td>351–352</td>
</tr>
<tr>
<td>modular debugger (See mdb debugger)</td>
<td></td>
</tr>
<tr>
<td>modulo operation</td>
<td>177–178</td>
</tr>
<tr>
<td>moving memory</td>
<td>338–339</td>
</tr>
<tr>
<td>MPI (Message Passing Interface), MPI_REDUCE function, MPSS</td>
<td>382–385, 383–384, 350</td>
</tr>
<tr>
<td>mpstat command</td>
<td>62–63</td>
</tr>
<tr>
<td>-mt compiler option</td>
<td>195</td>
</tr>
<tr>
<td>errno variable and, mtmalloc library</td>
<td>194, 195, 259</td>
</tr>
<tr>
<td>multiple data streams, optimizing use of, multiple execution pipes,</td>
<td>348–349</td>
</tr>
<tr>
<td>multiple page size support (MPSS), processors, in general</td>
<td></td>
</tr>
<tr>
<td>multiplication operations</td>
<td>9</td>
</tr>
<tr>
<td>multiply accumulate instructions</td>
<td>173–174</td>
</tr>
</tbody>
</table>
multithreading, 372, 385–402
atomic operations, 395–396
CMT systems. See CMT
data races, 412–413
debugging code, 413–416
false sharing, 397–399
memory layout, 399–402
mutexes, 389–395
parallelization. See parallelization
profiling multithreaded applications, 410–412
Pthreads, 385–387
role-based threads or processes, 445
sharing data between threads (example), 430–434
Thread Local Storage, 387–389
threads sharing a processor, 378
virtualization, 374–375
mutexes, 389–395, 430–432, 445
profiling performance, 410–412

N
named pipes, 380
NaN (Not-a-Number), 11, 155–157
comparisons, eliminating, 158
nested loops, 322–324
netstat command, 70
network activity, reporting, 70
nice value, process, 59
nm command, 87
no_side_effect pragma, 138–139, 426
ererrno variable and, 166
noalias pragma, 146
-noftstore compiler option, 101
nomemorydepend pragma, 142
Non-Uniform Memory Access (NUMA), 35, 36
nonvolatile variables, 97
.nt adornment (SPARC instruction), 25
NUMA (Non-Uniform Memory Access), 35, 36

O
-O compiler option, 98
O notation (complexity), 438–440
object files

combining with libraries. See libraries
and linking
disassembling, 89
segments in, reporting size of, 90
of_r_iu_req_mi_go counter, 310
OMP_NUM_THREADS environment
variable, 194, 406, 425
OpenMP API, 406
OpenMP specification, 194
debbug and, 264
load balancing, 429–430
parallelization, 402–403
eample of, 424–425
of loops, 403–406
sharing variables between threads,
432–434
operating system calls, reporting on, 80–81
Opteron processor performance counters,
310–317
optimization. See also performance
algorithmic complexity, 437–442
for CMT processors, 446
code layout optimizations, 107–116
crossfile optimization, 108–110
link time optimization, 115–116
compilation optimizations,
96–102, 116–123. See also Sun
Studio compiler, using
C- and C++-specific, 123–135
Fortran-specific, 135–136
including debug information and, 103
levels for, 93–95
data structures, 339–349
algorithmic complexity, 437–442
matrices and accesses, 347–348
multiple streams, 348–349
prefetching, 343–346
reorganizing, 339–343
various considerations, 346
floating-point. See floating-point
optimization
how to apply, 437–446
performance counters. See performance
counters
serial code, tuning, 442–444
serial vs. parallel applications, 418–419
of source code. See source code
optimizations
tail-call optimization and debug, 235–237
optimized maths library, 171
OR operation (logical), 9
ordered segments, 111
ordering memory (x64 processors), 46
$ORIGIN symbol, 186
out-of-order execution processors, 5–6
output registers, SPARC, 23, 26–27
register windows and, 27–28

P
-P compiler option, 250, 251
packet information, reporting on, 71
padding for thread data, 398
-pad=local compiler option, 101
pagesize command, 53–55
paging (memory), 17–18
changing what applications request, 54
reporting on, 57, 67
setting page size, 123
supported page size, reporting on, 53–55
thrashing in caches, page size and, 350–351
parallelization, 376–377, 444–446
automatic, 408–409, 425–429
example of, 417–434
using OpenMP, 402–407
loops, 402–403
section-based, 407
parentheses, honoring in floating-point
calculations, 158–159, 165
pargs command, 79
partitioning compute resources, 52
paths to libraries, specifying, 185–186
pbind command, 53
PC_MS_misses counter, 294
PC_port0_rd and PC_port1_rd counters,
293–294
PC_snoop_inv counter, 294
PC_soft_hit counter, 294
-pec compiler option, 271–272
peeling loops, 321–322
percentage sign (%) for SPARC registers, 23
performance, 437–446. See also
informational tools; optimization
algorithms and complexity, 437–442
branch mispredictions, 10, 107

performance counters for,
300–301, 315–316
counters. See performance counters
dynamic vs. static linking, 182
floating-point calculations. See also
floating-point optimization
integer maths, 174–178
Kahan Summation Formula, 161–163
reordering, 159–161
identifying consuming processes, 58–60
in-order vs. out-of-order processors, 5–6
load imbalance, 52, 429–430
mapfiles. See mapfiles
memory bandwidth. See bandwidth,
memory
memory latency. See cache latency
memory paging, 17
multithreaded applications. See also
multithreading
atomic operations, 395–396
false sharing, 397–399
mutexes, 393–396
optimizing for CMT processors, 446
parallelization. See parallelization
with prefetch. See prefetch instructions
processors, 5–6. See also process- and
processor-specific
profile feedback, 108, 111–115
profiling tools. See profiling performance
serial code, tuning, 442–444
structures, 346
subnormal number calculations, 154–155
system bus bandwidth, 15, 308
Performance Analyzer, 207–208
compiling for, 210
multithreaded applications, 410
performance counters, 218–219, 279–317
bus events, 76–77, 308
comparison with and without prefetch,
295–297
hardware events, reporting on, 73–76,
208, 304–305
Opteron processor, 310–317
reading, tools for, 279–281
SPARC64 VI processor, 309–310
TLB misses, 351–352
UltraSPARC III and IV processors,
281–302
performance counters (Continued)
  UltraSPARC IV and IV+ processors, 302–303
  UltraSPARC T1 processor, 304–308
  UltraSPARC T2 processor, 308–309
performance counters library, 196–197
pfiles command, 79
pgrep command, 61–62
physical memory, 16
pic0 and pic1 counters, 281
PID (process ID). See also processes of application, locating, 61–62
  arguments passed to, 79
  files held open by, reporting, 79
  spawned processes, 378–379
pins, CPU, 3
pipelines, 7, 320–321
  specifying safe degree of, 140–141
pipeloop pragma, 140–141
pipes, 7–11
  named, 380
  UltraSPARC III processors, 30–31
pmap command, 78–79
pointer aliasing in C and C++, 123–133
  diagnosing problems, 126
  loop invariant hoisting and, 324
  restricted pointers, 126–127
  specifying degree of, 127–133
pointer chasing, 336
pointers, restricted, 126–127, 443. See also -xalias_level compiler option
POSIX threads (Pthreads), 385–387
  memory layout, 399–402
  OpenMP specification vs., 402–403
  parallelization example, 422–424
  Thread Local Storage, 387–389
ppgsz command, 54
#pragma directives
  alias pragma, 144–145
  alias_level pragma, 143–144
  align pragma, 136–137
  does_not_read_global_data pragma, 137–138
  errno variable and, 166
  does_not_write_global_data pragma, 137–138
  errno variable and, 166
  fini pragma, 187
  init pragma, 187
  may_not_point_to pragma, 146–147
  may_point_to pragma, 145–146
  no_side_effect pragma, 138–139
    errno variable and, 166
  noalias pragma, 146
  nomemorydepend pragma, 141
  pipeloop pragma, 140–141
  rarely_called pragma, 139–140
  unroll pragma, 141–142
  pragmas, 136–142
  for aliasing control, 142–147
  predicting branches, 10. See also mispredicted branches
preferred page size, defining, 54
prefetch cache
  performance counters, 293–297
  UltraSPARC III and IV+ processors, 31, 33–34
prefetch instructions, 31, 116–118
  aggressiveness of prefetch insertion, 120.
    See also -xprefetch_level compiler option
  algorithmic complexity and, 441
  for cache lines, 335–337
  enabling prefetch generation, 118–119
  manual prefetch, 330–333
    store queue performance, 356–357
    structure prefetch, 346
  source code optimizations
    for integer data, 327–328
    with loop unrolling and pipelining, 320
    memory bandwidth and, 326–327
    storing data streams, 329
    speculative, number of, 101, 117
    structure prefetch, 343–346
preprocessing source code, 251
priority, process, 59
probe effect, 239
process- and processor-specific reporting
  bus activity, 76
  command line arguments, examining, 79
  files held open by processes, 79
  hardware performance counters, 73–76, 208, 304–305
  stack dumps, 79–80
  timing process execution, 72–73
  tracing application execution, 80–81
Index

trap activity, 77–78
user and system code activity, 82–84
virtual memory mapping, 78–79
process ID. See PID (process ID)
process resource utilization, reporting, 58–60
processes, 371–372
assigning roles to, 445
calls from, reporting on, 81
current, listing, 60–61
defined, 371
files held open by, 79
multiple, using, 378–385
cooperating processes, 378–382
copies of processes, 378
MPI (Message Passing Interface), 382–385
multithreaded. See multithreading
parallelization, 376–377, 444–446
automatic, 408–409, 425–429
example of, 417–434
using OpenMP, 402–407
processor activity, reporting all, 62–63
processor sets, controlling processor use with, 52–53
processor stall events, 299
processors, in general, 3–19, 371
caches. See caches
components of, 3–4
execution pipes, 7–11
named, 380
UltraSPARC III processors, 30–31
indexing and memory tagging, 18
instruction set architecture, 18–19
interacting with system, 14–16
multiple, 374–376
specifying with compiler,
99, 104, 105, 106–107
virtual memory, 16–18
profiling performance, 207–245. See also
optimization; performance
caller–callee information, 212–214
tail-call optimization and, 236–237
code coverage information
dtrace command, 241–244
tcov command, 239–241
collecting profiles, 208–210
command-line tool (er_print), 207, 214–215
compiler commentary, 244–245
with counters. See performance counters
interpreting profiles, 215–217
UltraSPARC processors, 217
mapfiles. See mapfiles
memory profiling across patterns
(dprofiling), 226–233
multithreaded applications, 410–412, 419
Performance Analyzer, about, 207–208
Performance Analyzer, compiling for, 210
profile feedback for compilation,
108, 111–115
profile information, gathering
dtrace command, 241–244
gprof command, 237–239
serial code, tuning, 442–444
spot tool, to generate reports, 223–226
tail-call optimization and debug, 235–237
viewing profiles with Analyzer GUI,
207, 210–212
program counter (x86 processors), 44
protocol, reporting activity for each, 70
prstat command, 58–60
prtconf command, 51
prtdiag command, 49–50
prtfru command, 51
prtpic1 command, 51
ps command, 60–61
psradm command, 51
psrinfo command, 51
psrset command, 52
pstack command, 79–80
.pt adornment (SPARC instruction), 25
Pthreads, 385–387
memory layout, 399–402
OpenMP specification vs., 402–403
parallelization example, 422–424
Thread Local Storage, 387–389
ptime command, 72–73
pvs command, 87–89

Q
quiet NaNs, 156–157

R
-R compiler option, 183, 185–186
race conditions, 389–392, 412–413, 434
rarely_called pragma, 139–140
RAW recycles, 352–354
Re_DC_miss counter, 287
Re_DC_missovhd counter, 287, 290
Re_EC_miss counter, 286, 287, 290
Re_L2_miss counter, 302
Re_L3_miss counter, 302
Re_RAW_miss counter, 299, 352–354
read misses
data cache, 283–285
instruction cache, 285
memory bandwidth measurements, 292–293
memory latency measurements
eample of, 288–290
synthetic metrics for, 290–292
second-level cache, 286
reads after writes, 352–354
reduction operations, 404–406
redundant floating-point calculations, 158–159
register windows, SPARC, 27–29
registers
fill and spill traps, 77
multiple data streams and, 349
SPARC architecture, 23–25
floating-point registers, 24, 29–30
integer registers, 26–27
unrolled loops and, 321
x64 architecture, 40, 43–45
regs command, 267–268
relative paths to libraries, specifying, 186
reordering floating-point calculations, 159–161
Kahan Summation Formula, 161–163
reorganizing data structures, 339–349
replacement algorithm (cache), 13
reporting on performance. See spot tool, to generate reports
reporting on system. See informational tools
RESTORE instruction (SPARC), 28–29
restricted pointers, 126–127, 443. See also -xalias_level compiler option
RISC (reduced instruction set computing), 19, 23
CISC vs., 41
Rogue Wave library, 198
role-specific threads, 445
rotation operations, 9
routines
call stack of, examining, 219–222
defined in files, identifying, 87
inlining, 108
copying or moving memory, 338–339
profile feedback for, 112
in libraries, learning how used, 189–191
library calls, 199–205
searching arrays with VIS instructions, 203–205
SIMD instructions and MediaLib, 202
for timing, 199–201
using most appropriate, 201–202
performance of. See optimization;
performance; profiling
performance
of standard libraries, compiler
recognition of, 133–135
time spent in, reporting, 211
RSS (resident set size), 58
Rstall_FP_use counter, 299
Rstall_IU_use counter, 299
Rstall_storeQ counter, 299, 355–357
runtime
application linking, information on, 191–192
optimization level and, 96
runtime array bounds checking, 259
runtime code checking, 256–262
runtime linker, 185
runtime stack overflow checking, 260–261
S
sar command, 64–68
SAVE instruction (SPARC), 27–28
SB_full counter, 304
multipliers for conversion to cycles, 306
scaling to multiple processors, 445
horizontal and vertical scaling, 375–376
using multiple processes, 378–385
scheduling processor instructions, 8–9, 104, 106
scoping symbols, 188
searching arrays with VIS instructions, 203–205
spawning processes, 378–379
speculative stride prediction, 336–337
spilling registers to memory, 28, 40
loop splitting and, 321–322
spill traps, 77
splitting loops, 322
spot tool, to generate reports, 223–226
src command, 215
SSE and SSE2 instruction set extensions (x64), 46. See also SIMD instructions
unavailable on 386 processor, 95–96
stack
default size, multithreading and, 399–400
default size, OpenMP and, 407–408
interpreting performance profiles for, 219–222
overflow checking, 260–261
placing local variables on, 135–136
stack dumps, 79–80
stack page size, specifying, 123
stack pointer (x86 code), 43, 44
stack space, 28–29
STACKSIZE environment variable, 260, 407
stalled cycles, 6, 316–317
RAW recycles, 352–354
store queue, 354–357
store queue, 354–357
stores (processors), 9
physical and virtual addresses of, 229
SPARC operations, 24
storing streams of data, 329
streams of data, storing, 329
strength reduction, 9, 325
stride predictor, 336–337
strong prefetches (UltraSPARC III and IV+), 33–34
structure pointers. See pointer aliasing in C and C++
structures. See data structures, optimizing
subblocked caches, 14
subdirectories. See directories
subexpressions, eliminating common, 324–325
subnormal numbers, 153–155
flushing to zero, 155
subtraction operations, 9
Sun HPC ClusterTools software, 384–385
Sun Math Library, 201–202
sun_prefetch_constructs, 330
Sun Studio compiler, using, 93–147
C and C++ pointer aliasing, 123–133
code layout optimizations, 107–116
compatibility with GCC, 147
debug information, generating, 102–103
optimization, 96–102
C- and C++-specific optimizations, 133–135
-fast option. See -fast compiler option
Fortran-specific optimizations, 135–136
general compiler optimizations, 116–123
-O compiler option, 98
volatile variables and, 94, 97
 pragmas, 136–142
in C, for aliasing control, 142–147
selecting target machine type, 103–107
types of compiler options, 93–95
-xtarget=generic option (x86), 95–96
Sun Studio Performance Analyzer. See profiling performance
superscalar processors, 7
swap command, 57–58
swap file usage
controlling, 57
reporting, 56, 57–58
swap file usage, reporting, 66
symbol scoping, 188
symbols in files, reporting on, 87
symmetric multiprocessing (SMP), 372
synchronization of processors, 15–16
system bandwidth consumption, 308
system buses, 15–16
system calls, reporting number of, 57, 66–67
system code activity, exploring, 82–84
system configuration reporting, 49–55
system libraries. See libraries and linking
system status reporting, 55–72
all processor activity, 62–63
current processes, listing, 60–61
disk space utilization, 71–72
I/O activity, 68–69
kernel statistics, 64–68
locating an application’s process ID, 61–62
network activity, 70
packet information, 71
process resource utilization, 58–60
swap file usage, 57–58
virtual memory utilization, 56–57
system time, reporting on, 57

T
T1 processor. See UltraSPARC T1 processor
T2 processor. See UltraSPARC T2 processor
tagging memory, 18
tail-call optimization and debug, 235–237
target machine type for compilation, 103
tcvo command, 239–241
.tcvo files, 240
third-level cache, 302–303
thrashing (caches), 12–13, 349–353
Thread Analyzer, 412
Thread Local Storage, 387–389
thread migrations, reporting on, 63
threads, 371–372. See also multithreading
assigning roles to, 445
parallelization. See parallelization
sharing data and variables, 430–434
sharing processor, 378
virtualization, 374–375
throughput computing, 378
tick counters, 199–200, 311
tiling loops, 323–324
time allocation, reporting on, 57
time-based profiling, 207–208
time command, 72–73
time function, 199
timex command, 72–73
timing functions, 199–201
timing harness (timing.h), 200–201
timing process execution, reporting, 72–73
TLB (Translation Lookaside Buffer), 17
events, performance counters for, 304, 306, 309, 314–315
layout, 107
multiple data streams and, 348
performance counter, 351–352
reporting supported TLB page sizes, 53–55
thrashing, 349–351
traps, 77
UltraSPARC III and IV+ processors, 33
tools. See developer tools; specific tool by name
tracing application execution, 80–81
tracing process execution, 81
tracking performance. See developer tools
training data for profile feedback, 113–114
transfers per second, reporting, 66
Translation Lookaside Buffer. See TLB
Translation Storage Buffer (TSB). See TLB
trap_DMMU_miss counter, 310
trap_IMMU_miss counter, 310
trips
caued by floating-point events, 166–167
to correct memory misalignment, 363–364
fill and spill traps, 77
reporting activity of, 77–78
TLB traps, 77
unfinished floating-point traps, 64
truss command, 80–81
TSB (Translation Storage Buffer). See TLB
TSO (Total Store Ordering), 36
tuning. See optimization
types (for variables), aliasing between. See aliasing control pragmas
U
-u compiler option, 253, 254
ulimit command, 260
UltraSPARC processors, 21–23. See also
SPARC architecture
interpreting performance profiles, 217
SPARC64 VI processor, 23, 38
performance counters, 309–310
UltraSPARC I processors, 22
UltraSPARC II processors, 22
UltraSPARC III processors, 22, 30–36
performance counters, 281–302
UltraSPARC IV and IV+ processors, 33
performance counters, 281–303
UltraSPARC T1 processor, 3–4, 22, 37
CMT (chip multithreading), 6
data cache, 13
MEMBAR instructions. See MEMBAR
instructions
page size, 17–18
performance counters, 304–308
UltraSPARC T2 processor, 22–23, 37–38
performance counters, 308–309
-xtarget=generic compiler option, 99
::umalog command (dbx), 274–276
umask flag, Opteron performance counters,
311
UMEM_DEBUG environment variable, 258
UMEM_LOGGING environment variable, 258
::umem_verify command (mdb), 274
uncoverage information, 225–226
unfinished floating-point traps, 64
unnecessary floating-point calculations, 158–159
unroll pragma, 141–142
unrolling loops, 320–321
for parallelization, 420–422
unrolling of loops, degree of, 141–142
user code activity, exploring, 82–84
user time, reporting on, 57

V
-v compiler compiler option, 248
V8 parameter passing (floating-point
functions), 178–180
V9 architecture (SPARC), 22, 30
variables
aliasing between. See aliasing control
pragmas
alignment of, specifying, 135, 136–137
global
link time optimization, 115
pointers and, 125–126
sharing between threads, 432–434
volatile, 94, 97
mutexes, 391–392, 432
VCXs (voluntary context switches), 60
vector library, 152
vectorizing floating-point computation,
151–153
versions of libraries
obtaining information on, 87–89
searching for instruction-set-specific, 186
vertical scaling, 375–376
vertical threading, 373
virtual addressing, 16
virtual memory, 16–18
mapping information, reporting for
process, 78–79
utilization of, reporting, 56–57
virtual processes, enabling, 51
virtual processors, 372, 373
virtualization, 374–375
VIS instructions (SPARC), 202–205. See also
SIMD instructions
for searching arrays, 203–205
vmstat command, 56–57
volatile variables, 94, 97
mutexes, 391–392, 432
voluntary context switches (VCXs), 60

W
-w compiler option, 252
+w and +w2 compiler options, 252
-w0 through -w4 compiler options, 253
watchmalloc.so library, 257–258
WC_miss counter, 297–298
WC_scrubbed counter, 297–298
WC_snoop_cb counter, 297–298
WC_wb_wo_read counter, 297–298
weak prefetches (UltraSPARC III and IV+), 33–34
whereami command, 266
worker threads. See Pthreads
write cache
  performance counters for, 297–298
  UltraSPARC III and IV+ processors, 32, 33
write misses
  data cache, 283–285
  instruction cache, 285
  memory bandwidth measurements, 292–293
  memory latency measurements
    example of, 288–290
    synthetic metrics for, 288–290
  second-level cache, 286

X
x64 architecture, 39–46
  byte ordering, 41–42
  instruction set extensions, 40, 46
  instruction template, 42–43
  ISA (information set architecture), 19
  memory ordering, 46
  as out-of-order processors, 6
  page size, 17, 123
  registers, 40, 43–45
  SPARC architecture vs., 41, 43, 46
  targeting for compilation, 103
  -xtarget=generic compiler option, 99
x86 processors, 39
  frame pointer (base pointer), 43, 44, 264
  -xtarget=generic compiler option, 95–96
x87 coprocessor, 46
-xalias_level compiler option, 127–133, 409
  -xalias_level= basic option, 129–130, 426
  -xar compiler option, 184
  -xarch compiler option, 104, 106–107
  -xarch=sparcfmaf option, 107, 174
  -xarch=sparcvis option, 107
  -xarch=sse option, 119
  -xarch=sse2 option, 152–153
  -xautopar compiler option, 408, 426
  -xbinopt compiler option, 225
  -xbinopt=prepare option, 261
  -xbuiltin compiler option, 133–135
    copying or moving memory, 338
    within -fast option, 101
    vectorized computation, 151
-xlibmil option and, 170–171
  errno variable, 166
-xcache compiler option, 104, 105
-xcheck=stkكف compiler option, 260–261, 408
-xchip compiler option, 104, 106
-xloopinfo compiler option, 404
-xcode compiler option, 185
-xcrossfile compiler option, 110
-xdebugformat compiler option, 262, 263
-xdepend compiler option, 120–121
  within -fast option, 101
-xdumpmacros compiler option, 252–253
-xe compiler option, 251
-xF compiler option, 110
-xhwprof compiler option, 226
-xinstrument compiler option, 412
-xipo compiler option, 110
-xlibmil compiler option, 170–171
  within -fast option, 101, 150
  -xbuiltin option and, 170–171
  errno variable, 166
-xlibmopt compiler option
  errno variable and, 166
  within -fast option, 101, 150, 171
-xlibmil= sunperf compiler option, 197
-xlinkopt compiler option, 115–116
-xlist compiler options (Fortran), 254–255
-xloopinfo compiler option, 408, 426
-xM compiler option, 250
-xmemalign compiler option, 121–123, 362
  within -fast option, 101
-xmemalign compiler option
  errno variable and, 166
-xm # optimization levels, 96–97
-xopenmp compiler option, 194, 264, 402, 413
-xpad compiler option, 135
-xpagesizer compiler option, 123
-xpagesize compiler option, 123
-xpagesize_heap compiler option, 123
-xpagesize_stack compiler option, 123
-xpg compiler option, 237
-xpost compiler option, 251
-xprefetch compiler option, 117, 326–327
-xprefetch_level compiler option,
  117, 118, 120
  within -fast option, 101
  manual prefetch, 330, 332
-xprofile compiler option
-xprofile=collect option, 111–112
-xprofile compiler option \textit{(Continued)}
  -xprofile=coverage option, 239
  -xprofile=use option, 112
-xreduction compiler option, 409
-xregs=frameptr compiler option, 264
  within -fast option, 101
-xrestrict compiler option, 127, 143
-\texttt{Xs} compiler mode, 171–172
-\texttt{xs} compiler option, 262, 263
-\texttt{xsfpc} compiler option, 172–173
-\texttt{xstackvar} compiler option, 135–136
-\texttt{Xt} compiler mode, 171–172
-\texttt{xtarget} compiler option
  -\texttt{xtarget=generic} option, 95–96, 99, 106
  -\texttt{xtarget=generic64} option, 106
  -\texttt{xtarget=native} option, 98, 99, 101
  -\texttt{xtarget=opteron} option, 105, 107
  -\texttt{xtarget=ultra3} option, 119, 330, 332
  -\texttt{xtarget_level=basic} compiler option, 101
-xtransition compiler option, 248
-xvector compiler option, 151–152
  within -fast option, 150
  -xvector=lib compiler option, 101
  -xvector=simd option, 152–153
-xvpara compiler option, 404–405, 425

Y
  \_\_global scoping specifier, 188
  \_\_hidden scoping specifier, 188
  \_\_MATHERR\_ERRNO\_DONTCARE \preprocessor{} variable, 165
  \_\_thread specifier, 388

Z
  zero, division by, 11
  handler for (example), 168–169
  zero divided by zero. \textit{See} NaN \textit{(Not-a-Number)}
Zones (Solaris Containers), 374–375
-ztext compiler option, 184–185