

# Solaris Application Programming

## Draft Table of Contents

*Darryl Gove*

<http://blogs.sun.com/d/>

|  |    |
|--|----|
| Chapter: 1 Preface.....                          | 10 |
| 1.1 About this book.....                         | 10 |
| 1.2 Goals and Assumptions .....                  | 10 |
| 1.3 Chapter Overview .....                       | 11 |
| 1.4 Acknowledgements.....                        | 11 |
| Chapter: 1 The generic processor .....           | 14 |
| 1.1 Chapter objectives.....                      | 14 |
| 1.2 The components of a processor .....          | 14 |
| 1.3 Clock speed.....                             | 15 |
| 1.4 Out-of-order processors .....                | 16 |
| 1.5 Chip Multi-Threading (CMT).....              | 16 |
| 1.6 Execution pipes.....                         | 17 |
| 1.6.1 Instruction latency.....                   | 18 |
| 1.6.2 Load/store pipe .....                      | 19 |
| 1.6.3 Integer operation pipe .....               | 19 |
| 1.6.4 Branch pipe .....                          | 19 |
| 1.6.5 Floating point pipe.....                   | 20 |
| 1.7 Caches.....                                  | 20 |
| 1.8 Interacting with the system.....             | 23 |
| 1.8.1 Bandwidth and latency.....                 | 23 |
| 1.8.2 System busses .....                        | 24 |
| 1.9 Virtual memory.....                          | 24 |
| 1.9.1 Overview.....                              | 24 |
| 1.9.2 TLBs and page size.....                    | 25 |
| 1.10 Indexing and tagging of memory.....         | 26 |
| 1.11 Instruction Set Architecture.....           | 26 |
| Chapter: 2 The SPARC family .....                | 28 |
| 2.1 Chapter objectives.....                      | 28 |
| 2.2 The UltraSPARC family.....                   | 28 |
| 2.2.1 History of the SPARC architecture.....     | 28 |
| 2.2.2 UltraSPARC processors.....                 | 28 |
| 2.3 The SPARC instruction set.....               | 29 |
| 2.3.1 A guide to the SPARC instruction set ..... | 29 |
| 2.3.2 Integer registers.....                     | 31 |
| 2.3.3 Register windows.....                      | 33 |
| 2.3.4 Floating point registers .....             | 35 |
| 2.4 32-bit and 64-bit code .....                 | 35 |
| 2.5 The UltraSPARC III family of processors..... | 36 |
| 2.5.1 The core of the CPU .....                  | 36 |
| 2.5.2 Communicating with memory .....            | 36 |

|   |    |
|---|----|
| 2.5.3 Prefetch .....  | 38 |
| 2.5.4 Blocking load on data cache miss.....   | 39 |
| 2.5.5 UltraSPARC III based systems.....   | 39 |
| 2.5.6 Total Store Order .....   | 41 |
| 2.6 UltraSPARC T1 .....   | 41 |
| 2.7 UltraSPARC T2 .....   | 42 |
| 2.8 SPARC64 VI.....   | 42 |
| Chapter: 3 The x64 family of processors.....  | 43 |
| 3.1 Chapter objectives.....   | 43 |
| 3.2 The x64 family of processors.....   | 43 |
| 3.3 The x86 processor - CISC and RISC .....   | 44 |
| 3.4 Byte ordering .....   | 44 |
| 3.5 Instruction template .....  | 45 |
| 3.6 Registers.....  | 46 |
| 3.7 Instruction set extensions and floating point .....   | 48 |
| 3.8 Memory ordering .....   | 48 |
| Chapter: 4 Informational tools.....   | 50 |
| 4.1 Chapter objectives.....   | 50 |
| 4.2 Tools which report system configuration .....   | 50 |
| 4.2.1 Introduction.....   | 50 |
| 4.2.2 Reporting general system information (prtdiag, prtconf, prtptcl, prtfpu).....             | 50 |
| 4.2.3 Enabling virtual processors (psrinfo & psradm).....                                       | 52 |
| 4.2.4 Controlling the use of processors through processor sets or binding (psrset, pbind) ..... | 52 |
| 4.2.5 Reporting instruction sets supported by hardware (isalist).....                           | 53 |
| 4.2.6 Reporting TLB page sizes supported by hardware (pagesize) .....                           | 54 |
| 4.2.7 Reporting summary of SPARC hardware characteristics (fpversion).....                      | 55 |
| 4.3 Tools which report current system status .....  | 56 |
| 4.3.1 Introduction.....   | 56 |
| 4.3.2 Reporting virtual memory utilisation (vmstat) .....                                       | 56 |
| 4.3.3 Reporting swap file usage (swap).....   | 57 |
| 4.3.4 Reporting process resource utilisation (prstat) .....                                     | 58 |
| 4.3.5 Listing processes (ps).....   | 60 |
| 4.3.6 Locating the Process ID of an application (pgrep) .....                                   | 61 |
| 4.3.7 Reporting activity for all processors (mpstat).....                                       | 61 |
| 4.3.8 Reporting kernel statistics (kstat) .....   | 63 |
| 4.3.9 Generating a report of system activity (sar).....   | 64 |
| 4.3.10 Reporting I/O activity (iostat).....   | 67 |
| 4.3.11 Reporting network activity (netstat) .....   | 69 |
| 4.3.12 snoop.....   | 70 |
| 4.3.13 Reporting disk space utilisation (df).....   | 70 |
| 4.3.14 Reporting disk space used by files (du).....   | 71 |
| 4.4 Process and processor specific tools.....   | 71 |
| 4.4.1 Introduction.....   | 71 |
| 4.4.2 Timing process execution (time, timex, ptime).....  | 71 |
| 4.4.3 Reporting system-wide hardware counter activity (cpustat) .....                           | 72 |

|   |  |     |
|---|--|-----|
| track)  | 4.4.4 Reporting hardware performance counter activity for a single process (cpu-74     |     |
|   | 4.4.5 Reporting bus activity (busstat) .....   | 75  |
|   | 4.4.6 Reporting on trap activity (trapstat) .....                                      | 75  |
|   | 4.4.7 Reporting virtual memory mapping information for a process (pmap) .....          | 77  |
|   | 4.4.8 Examining commandline arguments passed to process (pargs) .....                  | 77  |
|   | 4.4.9 Reporting the files held open by a process (pfiles) .....                        | 78  |
|   | 4.4.10 Examining current stack of process (pstack) .....                               | 78  |
|   | 4.4.11 Tracing application execution (truss) .....                                     | 78  |
|   | 4.4.12 Exploring user code and kernel activity with dtrace .....                       | 80  |
|   | 4.5 Information about applications .....   | 83  |
|   | 4.5.1 Reporting library linkage (ldd) .....  | 83  |
|   | 4.5.2 Reporting type of contents held in a file (file) .....                           | 85  |
|   | 4.5.3 Reporting symbols in a file (nm) .....   | 85  |
|   | 4.5.4 Reporting library version information (pvs) .....                                | 86  |
|   | 4.5.5 Examining the disassembly of an application, library, or object (dis) .....      | 87  |
| (size)  | 4.5.6 Reporting the size of the various segments in an application, library, or object | 88  |
| mcs)  | 4.5.7 Reporting meta-data held in a file (dumpstabs, dwarfdump, elfdump, dump,         | 88  |
| Chapter: 5 Using the compiler .....   |  | 92  |
| 5.1 Chapter objectives .....  |  | 92  |
| 5.2 Three sets of compiler options .....  |  | 92  |
| 5.3 Using -xtarget=generic on x86 .....   |  | 94  |
| 5.4 Optimisation .....  |  | 94  |
| 5.4.1 Optimisation levels .....   |  | 94  |
| 5.4.2 Using the -O optimisation flag .....                                      |  | 96  |
| 5.4.3 Using the -fast compiler flag .....                                       |  | 96  |
| 5.4.4 Specifying architecture with -fast .....                                  |  | 97  |
| 5.4.5 Deconstructing -fast .....  |  | 97  |
| 5.4.6 Performance optimisations in -fast (for the Sun Studio 12 compiler) ..... |  | 98  |
| 5.5 Generating debug information .....  |  | 100 |
| 5.5.1 Debug information flags .....   |  | 100 |
| 5.5.2 Debug and optimisation .....  |  | 100 |
| 5.6 Selecting the target machine type for an application .....                  |  | 101 |
| 5.6.1 Choosing between 32-bit and 64-bit applications .....                     |  | 101 |
| 5.6.2 The generic target .....  |  | 101 |
| 5.6.3 Specifying cache configuration using the -xcache flag .....               |  | 102 |
| 5.6.4 Specifying code scheduling using the -xchip flag .....                    |  | 103 |
| 5.6.5 The -xarch flag and -m32/-m64 .....                                       |  | 103 |
| 5.7 Code layout optimisations .....   |  | 104 |
| 5.7.1 Introduction .....  |  | 104 |
| 5.7.2 Crossfile optimisation .....  |  | 105 |
| 5.7.3 Mapfiles .....  |  | 107 |
| 5.7.4 Profile feedback .....  |  | 108 |
| 5.7.5 Link time optimisation .....  |  | 111 |

|     |  |     |
|-----|--|-----|
|     | 5.8 General compiler optimisations .....   | 112 |
|     | 5.8.1 Prefetch instructions .....  | 112 |
|     | 5.8.2 Enabling prefetch generation (-xprefetch).....                               | 114 |
|     | 5.8.3 Controlling the aggressiveness of prefetch insertion (-xprefetch_level)....  | 116 |
|     | 5.8.4 Enabling dependence analysis (-xdepend).....                                 | 116 |
|     | 5.8.5 Handling misaligned memory accesses on SPARC (-xmalign /-dalign).....        |     |
| 117 | 5.8.6 Setting page size using -xpagesize=<size>.....                               | 118 |
|     | 5.9 Pointer aliasing in C and C++.....   | 119 |
|     | 5.9.1 The problem with pointers.....   | 119 |
|     | 5.9.2 Diagnosing aliasing problems.....  | 121 |
|     | 5.9.3 Using restricted pointers in C and C++ to reduce aliasing issues.....        | 122 |
|     | 5.9.4 Using the -xalias_level flag to specify the degree of pointer aliasing ..... | 123 |
|     | 5.9.5 -xalias_level for C.....   | 124 |
|     | 5.9.6 -xalias_level=any .....  | 124 |
|     | 5.9.7 -xalias_level=basic in C.....  | 125 |
|     | 5.9.8 -xalias_level=weak in C.....   | 126 |
|     | 5.9.9 -xalias_level=layout in C.....   | 127 |
|     | 5.9.10 -xalias_level=strict in C.....  | 128 |
|     | 5.9.11 -xalias_level=std in C.....   | 128 |
|     | 5.9.12 -xalias_level=strong in C.....  | 128 |
|     | 5.9.13 xalias_level in C++.....  | 129 |
|     | 5.9.14 -xalias_level=simple in C++.....  | 129 |
|     | 5.9.15 -xalias_level=compatible in C++.....  | 129 |
|     | 5.10 Other C and C++ specific compiler optimisations.....                          | 129 |
|     | 5.10.1 Enabling the recognition of standard library routines (-xbuiltin) .....     | 129 |
|     | 5.11 Fortran specific compiler optimisations .....                                 | 130 |
|     | 5.11.1 Aligning variables for optimal layout (-xpad) .....                         | 130 |
|     | 5.11.2 Placing local variables on the stack (-xstackvar).....                      | 131 |
|     | 5.12 Compiler pragmas.....   | 131 |
|     | 5.12.1 Introduction.....   | 131 |
|     | 5.12.2 Specifying alignment of variables .....                                     | 131 |
|     | 5.12.3 Specifying a functions access to global data .....                          | 132 |
|     | 5.12.4 Specifying that a function has no side-effects .....                        | 134 |
|     | 5.12.5 Specifying that a function is infrequently called .....                     | 135 |
|     | 5.12.6 Specifying safe degree of pipelining for a particular loop.....             | 136 |
|     | 5.12.7 Specifying that a loop has no memory dependencies within a single iteration |     |
| 137 | 5.12.8 Specifying the degree of loop unrolling .....                               | 137 |
|     | 5.13 Using pragmas in C for finer aliasing control.....                            | 138 |
|     | 5.13.1 Asserting the degree of aliasing between variables.....                     | 139 |
|     | 5.13.2 Asserting that variables do alias .....                                     | 140 |
|     | 5.13.3 Asserting aliasing with non-pointer variables .....                         | 141 |
|     | 5.13.4 Asserting that variables do not alias .....                                 | 141 |
|     | 5.13.5 Asserting no aliasing with non-pointer variables .....                      | 142 |
|     | 5.14 Compatibility with GCC.....   | 143 |

|   |     |
|---|-----|
| Chapter: 6 Floating point optimisation .....  | 144 |
| 6.1 Chapter objectives.....   | 144 |
| 6.2 Floating point optimisation flags .....   | 144 |
| 6.2.1 Mathematical optimisations in -fast.....                                      | 144 |
| 6.2.2 IEEE-754 and floating point.....  | 145 |
| 6.2.3 Vectorising floating point computation (-xvector) .....                       | 145 |
| 6.2.4 Vectorising computation using SIMD instructions (-xvector=simd)(x64 only)     |     |
| 147   |     |
| 6.2.5 Subnormal numbers .....   | 147 |
| 6.2.6 Flushing subnormal numbers to zero (-fns).....                                | 149 |
| 6.2.7 Handling values that are not a number .....                                   | 149 |
| 6.2.8 Enabling floating point expression simplification (-fsimple).....             | 150 |
| 6.2.9 Elimination of comparisons .....  | 151 |
| 6.2.10 Elimination of unnecessary calculation .....                                 | 152 |
| 6.2.11 Reordering of calculations .....   | 152 |
| 6.2.12 Kahan Summation Formula .....  | 155 |
| 6.2.13 Hoisting of divides.....   | 157 |
| 6.2.14 Honouring of parentheses at levels of floating point simplification.....     | 159 |
| 6.2.15 Effect of -fast on errno.....  | 159 |
| 6.2.16 Specifying which floating point events cause traps (-ftrap) .....            | 160 |
| 6.2.17 The floating point exception flags .....                                     | 160 |
| 6.2.18 Floating point exceptions in C99 .....                                       | 162 |
| 6.2.19 Using inline template versions of floating point functions (-xlibmil) .....  | 163 |
| 6.2.20 Using the optimised maths library (-xlibmopt).....                           | 164 |
| 6.2.21 Do not promote single precision values to double precision (-fsingle for C). |     |
| 165   |     |
| 6.2.22 Store floating point constants in single precision (-xsfpcnst for C) .....   | 165 |
| 6.3 Floating point multiply accumulate instructions .....                           | 166 |
| 6.4 Integer maths.....  | 167 |
| 6.4.1 Other integer maths opportunities.....  | 170 |
| 6.5 Floating point parameter passing with SPARC V8 code.....                        | 171 |
| Chapter: 7 Libraries and linking .....  | 174 |
| 7.1 Introduction.....   | 174 |
| 7.2 Linking.....  | 174 |
| 7.2.1 Overview of linking.....  | 174 |
| 7.2.2 Dynamic and static linking .....  | 174 |
| 7.2.3 Linking libraries.....  | 175 |
| 7.2.4 Creating a static library.....  | 176 |
| 7.2.5 Creating a dynamic library .....  | 176 |
| 7.2.6 Specifying the location of libraries.....                                     | 177 |
| 7.2.7 Lazy loading of libraries.....  | 178 |
| 7.2.8 Initialisation and finalisation code in libraries.....                        | 178 |
| 7.2.9 Symbol scoping.....   | 179 |
| 7.2.10 library interposition.....   | 181 |
| 7.2.11 Using the debug interface .....  | 183 |
| 7.2.12 Using the audit interface .....  | 184 |

|   |     |
|---|-----|
| 7.3 Libraries of interest .....   | 185 |
| 7.3.1 The C runtime library (libc and libc_psr) .....                   | 185 |
| 7.3.2 Memory management libraries .....                                 | 185 |
| 7.3.3 libfast .....   | 188 |
| 7.3.4 The performance library .....                                     | 188 |
| 7.3.5 STLport4 .....  | 189 |
| 7.4 Library calls .....   | 190 |
| 7.4.1 Library routines for timing .....                                 | 190 |
| 7.4.2 Picking the most appropriate library routines .....               | 193 |
| 7.4.3 SIMD instructions and the Media Library .....                     | 194 |
| 7.4.4 Searching arrays using VIS instructions .....                     | 194 |
| Chapter: 8 Performance profiling tools .....                            | 198 |
| 8.1 Introduction .....  | 198 |
| 8.2 The Sun Studio Performance Analyzer .....                           | 198 |
| 8.3 Collecting profiles .....   | 199 |
| 8.4 Compiling for the Performance Analyzer .....                        | 200 |
| 8.5 Viewing profiles using the GUI .....                                | 200 |
| 8.6 Caller-callee information .....                                     | 203 |
| 8.7 Using the command line tool for performance analysis .....          | 204 |
| 8.8 Interpreting profiles .....   | 205 |
| 8.9 Intepreting profiles from UltraSPARC III/IV processors .....        | 207 |
| 8.10 Profiling using performance counters .....                         | 207 |
| 8.11 Interpreting call stacks .....                                     | 208 |
| 8.12 Generating mapfiles .....  | 212 |
| 8.13 Generating reports on performance using spot .....                 | 212 |
| 8.14 Profiling memory access patterns .....                             | 215 |
| 8.15 er_kernel .....  | 224 |
| 8.16 Tail-call optimisation and debug .....                             | 227 |
| 8.17 Gathering profile information using gprof .....                    | 228 |
| 8.18 Using tcov to get code coverage information .....                  | 230 |
| 8.19 Using dtrace to gather profile and coverage information .....      | 232 |
| 8.20 Compiler commentary .....  | 234 |
| Chapter: 9 Correctness and debug .....                                  | 237 |
| 9.1 Introduction .....  | 237 |
| 9.2 Compile time checking .....   | 237 |
| 9.2.1 Introduction .....  | 237 |
| 9.2.2 Compile time checking for C source code .....                     | 237 |
| 9.2.3 Checking of C source code using lint .....                        | 237 |
| 9.2.4 Source processing options common to the C and C++ compilers ..... | 239 |
| 9.2.5 C++ .....   | 241 |
| 9.2.6 Fortran .....   | 242 |
| 9.3 Runtime checking .....  | 245 |
| 9.3.1 Bounds checking .....   | 245 |
| 9.3.2 Watchmalloc .....   | 246 |
| 9.3.3 Debugging options under other mallocs .....                       | 247 |
| 9.3.4 Runtime array bounds checking in Fortran .....                    | 247 |

|   |     |
|---|-----|
| 9.3.5 Runtime stack overflow checking.....                                | 248 |
| 9.3.6 Memory access error detection using discover.....                   | 250 |
| 9.4 Debugging using dbx.....  | 251 |
| 9.4.1 Debug compiler flags.....   | 251 |
| 9.4.2 Debug and optimisation.....   | 252 |
| 9.4.3 Debug information format.....                                       | 252 |
| 9.4.4 Debug and OpenMP.....   | 252 |
| 9.4.5 Frame pointer optimisation on x86.....                              | 253 |
| 9.4.6 Running the debugger on a core file.....                            | 253 |
| 9.4.7 Example of debugging an application.....                            | 253 |
| 9.4.8 Running an application under dbx.....                               | 256 |
| 9.5 Locating optimisation bugs using ATS.....                             | 259 |
| 9.6 Debugging using mdb.....  | 262 |
| Chapter: 10 Performance counter metrics.....                              | 267 |
| 10.1 Chapter objectives.....  | 267 |
| 10.2 Reading the performance counters.....                                | 267 |
| 10.3 UltraSPARC III and UltraSPARC IV performance couners.....            | 268 |
| 10.3.1 Instructions and cycles.....                                       | 268 |
| 10.3.2 Data cache events.....   | 270 |
| 10.3.3 Instruction cache events.....                                      | 272 |
| 10.3.4 Second level cache events.....                                     | 273 |
| 10.3.5 Cycles lost to cache miss events.....                              | 273 |
| 10.3.6 Example of cache access metrics.....                               | 274 |
| 10.3.7 Synthetic metrics for latency.....                                 | 277 |
| 10.3.8 Synthetic metrics for memory bandwidth consumption.....            | 279 |
| 10.3.9 Prefetch cache events.....   | 280 |
| 10.3.10 Comparison of performance counters with and without prefetch..... | 283 |
| 10.3.11 Write cache events.....   | 284 |
| 10.3.12 Cycles lost to processor stall events.....                        | 285 |
| 10.3.13 Branch misprediction.....   | 286 |
| 10.3.14 Memory controller events.....                                     | 287 |
| 10.4 Performance counters on the UltraSPARC IV and UltraSPARC IV+.....    | 288 |
| 10.4.1 Introduction.....  | 288 |
| 10.4.2 UltraSPARC IV+ L3 cache.....                                       | 289 |
| 10.4.3 Memory controller events.....                                      | 289 |
| 10.5 Performance counters on the UltraSPARC T1.....                       | 290 |
| 10.5.1 Hardware performance counters.....                                 | 290 |
| 10.5.2 UltraSPARC T1 cycle budget.....                                    | 291 |
| 10.5.3 Performance counters at the core level.....                        | 293 |
| 10.5.4 Calculating system bandwidth consumption.....                      | 294 |
| 10.6 UltraSPARC T2 performance counters.....                              | 294 |
| 10.7 SPARC64 VI performance counters.....                                 | 296 |
| 10.8 Opteron performance counters.....                                    | 296 |
| 10.8.1 Introduction.....  | 296 |
| 10.8.2 Instructions.....  | 297 |
| 10.8.3 Instruction cache events.....                                      | 298 |

|   |     |
|---|-----|
| 10.8.4 Data cache events.....                                       | 299 |
| 10.8.5 TLB events.....  | 300 |
| 10.8.6 Branches events .....  | 301 |
| 10.8.7 Stall cycles .....   | 302 |
| Chapter: 11 Source code optimisations .....                         | 305 |
| 11.1 Overview.....  | 305 |
| 11.2 Traditional optimisations .....                                | 305 |
| 11.2.1 Introduction.....  | 305 |
| 11.2.2 Loop unrolling and pipelining .....                          | 305 |
| 11.2.3 Loop peeling, fusion, and splitting .....                    | 306 |
| 11.2.4 Loop interchange and tiling.....                             | 307 |
| 11.2.5 Loop invariant hoisting.....                                 | 309 |
| 11.2.6 Common sub-expression elimination .....                      | 310 |
| 11.2.7 Strength reduction.....                                      | 310 |
| 11.2.8 Function cloning .....                                       | 310 |
| 11.3 Data locality, bandwidth and latency.....                      | 311 |
| 11.3.1 Bandwidth.....   | 311 |
| 11.3.2 Integer data .....   | 312 |
| 11.3.3 Storing streams .....  | 314 |
| 11.3.4 Manual prefetch.....   | 314 |
| 11.3.5 Latency.....   | 318 |
| 11.3.6 Copying and moving memory .....                              | 323 |
| 11.4 Data structures .....  | 324 |
| 11.4.1 Structure reorganising.....                                  | 324 |
| 11.4.2 Structure prefetching.....                                   | 328 |
| 11.4.3 Considerations for optimal performance from structures ..... | 331 |
| 11.4.4 Matrices and accesses .....                                  | 332 |
| 11.4.5 Multiple streams .....                                       | 334 |
| 11.5 Thrashing .....  | 335 |
| 11.5.1 Summary.....   | 335 |
| 11.5.2 DTLB performance counter.....                                | 337 |
| 11.6 Reads after writes.....  | 338 |
| 11.7 Store queue .....  | 341 |
| 11.7.1 Stalls.....  | 341 |
| 11.7.2 Detecting store queue stalls .....                           | 341 |
| 11.8 If statements .....  | 343 |
| 11.8.1 Introduction.....  | 343 |
| 11.8.2 Conditional moves .....                                      | 344 |
| 11.8.3 Misaligned memory accesses on SPARC processors .....         | 347 |
| 11.9 File handling in 32-bit applications .....                     | 351 |
| 11.9.1 File descriptor limits .....                                 | 351 |
| 11.9.2 Handling large files in 32-bit applications.....             | 352 |
| Chapter: 12 Multi-core, multi-process, multi-thread .....           | 356 |
| 12.1 Introduction.....  | 356 |
| 12.2 Processes, threads, processors, cores, and CMT.....            | 356 |
| 12.3 Virtualisation .....   | 358 |

|  |     |
|--|-----|
| 12.4 Horizontal and vertical scaling .....                     | 359 |
| 12.5 Parallelisation.....                                      | 359 |
| 12.6 Scaling using multiple processes .....                    | 361 |
| 12.6.1 Multiple processes .....                                | 361 |
| 12.6.2 Multiple co-operating processes .....                   | 362 |
| 12.6.3 Parallelism using MPI.....                              | 366 |
| 12.7 Multithreaded applications .....                          | 369 |
| 12.7.1 Parallelisation using Pthreads .....                    | 369 |
| 12.7.2 Thread Local Storage.....                               | 371 |
| 12.7.3 Mutexes.....  | 373 |
| 12.7.4 Using atomic operations .....                           | 380 |
| 12.7.5 False sharing .....                                     | 382 |
| 12.7.6 Memory layout for a threaded application.....           | 385 |
| 12.8 Parallelising applications using OpenMP .....             | 388 |
| 12.9 Using OpenMP directives to parallelise loops.....         | 389 |
| 12.10 Using the OpenMP API .....                               | 392 |
| 12.11 Parallel sections .....                                  | 393 |
| 12.11.1 Setting stack sizes for OpenMP .....                   | 394 |
| 12.12 Automatic parallelisation of applications .....          | 394 |
| 12.13 Profiling multithreaded applications.....                | 396 |
| 12.14 Detecting data races in multithreaded applications ..... | 398 |
| 12.15 Debugging multithreaded codes .....                      | 399 |
| 12.16 Parallelising a serial application .....                 | 403 |
| 12.16.1 Example application .....                              | 403 |
| 12.16.2 Impact of optimisation on serial performance .....     | 405 |
| 12.16.3 Profiling the serial application.....                  | 406 |
| 12.16.4 Unrolling the critical loop.....                       | 407 |
| 12.16.5 Parallelising using Pthreads .....                     | 410 |
| 12.16.6 Parallelisation using OpenMP .....                     | 412 |
| 12.16.7 Autoparallelisation.....                               | 413 |
| 12.16.8 Load balancing with OpenMP .....                       | 416 |
| 12.16.9 Sharing data between threads .....                     | 417 |
| 12.16.10 Sharing variables between threads using OpenMP.....   | 419 |
| Chapter: 13 Performance analysis .....                         | 424 |
| 13.1 Introduction.....   | 424 |
| 13.2 Algorithms and complexity .....                           | 424 |
| 13.3 Tuning serial codes .....                                 | 428 |
| 13.4 Exploring parallelism.....                                | 430 |
| 13.5 Optimising for CMT processors .....                       | 431 |