

Listings

| | | |
|------|--|-----|
| 1.1 | Add two vectors in C, with implied serial ordering | 17 |
| 1.2 | Overlapping (aliased) arguments in C..... | 17 |
| 1.3 | Add two vectors using Cilk Plus array notation | 17 |
| 1.4 | An ordered sum creates a dependence in C | 18 |
| 1.5 | A parallel sum, expressed as a reduction operation in Cilk Plus..... | 18 |
| 1.6 | Function calls with step-by-step ordering specified in C | 18 |
| 1.7 | Function calls with no required ordering in Cilk Plus | 19 |
| 1.8 | Serial vector addition coded as a loop in C | 24 |
| 1.9 | Parallel vector addition using Cilk Plus..... | 25 |
| 1.10 | Parallel vector addition using ArBB | 25 |
| 1.11 | Scalar function for addition in C | 26 |
| 1.12 | Vectorized function for addition in Cilk Plus | 26 |
| 1.13 | Serial Fibonacci computation in C | 31 |
| 1.14 | Parallel Cilk Plus variant of Listing 1.13 | 31 |
| 1.15 | Vector computation in ArBB | 34 |
| 1.16 | Elemental function computation in ArBB | 35 |
| 3.1 | Serial sequence in pseudocode | 82 |
| 3.2 | Serial sequence, second example, in pseudocode | 83 |
| 3.3 | Serial selection in pseudocode..... | 84 |
| 3.4 | Iteration using a <code>while</code> loop in pseudocode | 85 |
| 3.5 | Iteration using a <code>for</code> loop in pseudocode | 85 |
| 3.6 | Demonstration of <code>while</code> / <code>for</code> equivalence in pseudocode | 86 |
| 3.7 | A difficult example in C | 87 |
| 3.8 | Another difficult example in C | 87 |
| 3.9 | Serial implementation of reduction..... | 91 |
| 3.10 | Serial implementation of scan | 93 |
| 3.11 | Superscalar sequence in pseudocode | 103 |
| 4.1 | Serial implementation of SAXPY in C | 126 |
| 4.2 | Tiled implementation of SAXPY in TBB..... | 126 |
| 4.3 | SAXPY in Cilk Plus using <code>cilk_for</code> | 127 |
| 4.4 | SAXPY in Cilk Plus using <code>cilk_for</code> and array notation for explicit vectorization | 127 |
| 4.5 | SAXPY in OpenMP..... | 128 |
| 4.6 | SAXPY in ArBB, using a vector expression..... | 129 |

| | |
|---|-----|
| 4.7 SAXPY in ArBB, using binding code for vector expression implementation | 129 |
| 4.8 SAXPY in ArBB, using an elemental function | 130 |
| 4.9 SAXPY in ArBB, call operation | 130 |
| 4.10 SAXPY in OpenCL kernel language | 131 |
| 4.11 Serial implementation of Mandelbrot in C | 133 |
| 4.12 Tiled implementation of Mandelbrot in TBB | 133 |
| 4.13 Mandelbrot using <code>cilk_for</code> in Cilk Plus | 134 |
| 4.14 Mandelbrot in Cilk Plus using <code>cilk_for</code> and array notation for explicit vectorization | 135 |
| 4.15 Mandelbrot in OpenMP | 136 |
| 4.16 Mandelbrot elemental function for ArBB map operation | 136 |
| 4.17 Mandelbrot call code for ArBB implementation | 137 |
| 4.18 Mandelbrot binding code for ArBB implementation | 137 |
| 4.19 Mandelbrot kernel code for OpenCL implementation | 138 |
| 5.1 Serial reduction in C++ for 0 or more elements | 146 |
| 5.2 Serial reduction in C++ for 1 or more elements | 147 |
| 5.3 Serial implementation of dot product in C++ | 155 |
| 5.4 Vectorized dot product implemented using SSE intrinsics | 156 |
| 5.5 Dot product implemented in TBB | 157 |
| 5.6 Modification of Listing 5.5 with double-precision operations | 158 |
| 5.7 Dot product implemented in Cilk Plus array notation | 159 |
| 5.8 Dot product implementation in Cilk Plus using explicit tiling | 159 |
| 5.9 Modification of Listing 5.8 with double-precision operations for multiplication and accumulation | 160 |
| 5.10 Dot product implemented in OpenMP | 160 |
| 5.11 Dot product implemented in ArBB | 161 |
| 5.12 Dot product implementation in ArBB, wrapper code | 161 |
| 5.13 High-precision dot product implemented in ArBB | 162 |
| 5.14 Serial implementation of inclusive scan in C++ | 163 |
| 5.15 Serial implementation of exclusive scan in C++ | 164 |
| 5.16 Three-phase tiled implementation of a scan in OpenMP | 168 |
| 5.17 Serial integrated table preparation in C++ | 171 |
| 5.18 Generic test function for integration | 171 |
| 5.19 Concrete instantiation of test function for integration | 171 |
| 5.20 Serial implementation of integrated table lookup in C++ | 172 |
| 5.21 Integrated table preparation in Cilk Plus | 173 |
| 5.22 Integrated table preparation in TBB | 174 |
| 5.23 Integrated table preparation in ArBB | 175 |
| 5.24 Integrated table lookup in ArBB | 176 |

| | | |
|------|---|-----|
| 6.1 | Serial implementation of gather in pseudocode | 180 |
| 6.2 | Serial implementation of scatter in pseudocode | 186 |
| 6.3 | Array of structures (AoS) data organization | 194 |
| 6.4 | Structure of arrays (SoA) data organization | 195 |
| 7.1 | Serial implementation of stencil | 200 |
| 7.2 | Serial 2D recurrence | 205 |
| 8.1 | Recursive implementation of the map pattern in Cilk Plus | 215 |
| 8.2 | Modification of Listing 8.1 that changes tail call into a goto | 217 |
| 8.3 | Cleaned-up semi-recursive map in Cilk Plus | 217 |
| 8.4 | Three loop forms illustrating <i>steal-continuation</i> versus <i>steal-child</i> | 219 |
| 8.5 | Flat algorithm for polynomial multiplication using Cilk Plus array notation | 224 |
| 8.6 | Karatsuba multiplication in Cilk Plus | 225 |
| 8.7 | Type for scratch space | 227 |
| 8.8 | Pseudocode for recursive matrix multiplication | 229 |
| 8.9 | Code shared by Quicksort implementations | 232 |
| 8.10 | Fully recursive parallel Quicksort using Cilk Plus | 233 |
| 8.11 | Semi-recursive parallel Quicksort using Cilk Plus | 234 |
| 8.12 | Semi-iterative parallel Quicksort using TBB | 235 |
| 8.13 | Quicksort in TBB that achieves Cilk Plus space guarantee | 236 |
| 8.14 | Recursive implementation of parallel reduction in Cilk Plus | 238 |
| 8.15 | Using a hyperobject to avoid a race in Cilk Plus | 239 |
| 8.16 | Using a local reducer in Cilk Plus | 241 |
| 8.17 | Top-level code for tiled parallel scan..... | 243 |
| 8.18 | Upsweep phase for tiled parallel scan in Cilk Plus | 244 |
| 8.19 | Downsweep phase for tiled parallel scan in Cilk Plus..... | 244 |
| 8.20 | Implementing pack pattern with <code>cilk_scan</code> from Listing 8.17 | 245 |
| 8.21 | Base case for evaluating a diamond of lattice points | 249 |
| 8.22 | Code for parallel recursive evaluation of binomial lattice in Cilk Plus..... | 250 |
| 8.23 | Marching over diamonds in Cilk Plus..... | 251 |
| 9.1 | Serial implementation of a pipeline | 257 |
| 9.2 | Pipeline in TBB | 258 |
| 9.3 | Pipeline in Cilk Plus equivalent to the serial pipeline in Listing 9.1 | 259 |
| 9.4 | Defining a reducer for serializing consumption of items in Cilk Plus | 260 |
| 10.1 | Serial code for simulating wavefield | 268 |
| 10.2 | Code for one-dimensional iterated stencil | 271 |
| 10.3 | Base case for applying stencil to space-time trapezoid | 273 |
| 10.4 | Parallel cache-oblivious trapezoid decomposition in Cilk Plus | 274 |
| 10.5 | ArBB code for simulating a wavefield | 276 |

| | |
|--|-----|
| 11.1 K-means clustering in Cilk Plus | 282 |
| 11.2 Type <code>sum_and_count</code> for computing mean of points in a cluster | 283 |
| 11.3 Defining a hyperobject for summing an array elementwise in Cilk Plus | 284 |
| 11.4 Declaring a type <code>tls_type</code> for thread-local views in TBB | 285 |
| 11.5 Walking local views to detect changes | 286 |
| 11.6 Walking local views to accumulate a global sum..... | 286 |
| 11.7 Routine for finding index of centroid closest to a given point | 287 |
| 11.8 K-means clustering in TBB | 288 |
| 12.1 Declarations for bzip2 pipeline | 294 |
| 12.2 Use of TBB <code>parallel_pipeline</code> to coordinate bzip2 actions | 295 |
| 12.3 Sketch of bzip2 pipeline in Cilk Plus using a consumer reducer | 297 |
| 13.1 Serial merge | 300 |
| 13.2 Parallel merge in Cilk Plus | 301 |
| 13.3 Converting parallel merge from Cilk Plus to TBB | 302 |
| 13.4 Parallel merge sort in Cilk Plus..... | 304 |
| 14.1 Top-level code for parallel sample sort | 308 |
| 14.2 Code for mapping keys to bins | 310 |
| 14.3 Parallel binning of keys using Cilk Plus | 311 |
| 14.4 Repacking and subsorting using Cilk Plus | 312 |
| 14.5 Using Cilk Plus to move and destroy a sequence, without an explicit loop! | 313 |
| 15.1 Recursive Cholesky decomposition | 318 |
| 15.2 Parallel triangular solve in Cilk Plus | 320 |
| 15.3 Parallel symmetric rank update in Cilk Plus | 321 |
| 15.4 Converting parallel symmetric rank update from Cilk Plus to TBB | 322 |
| B.1 Simple example use of <code>cilk_for</code> | 332 |
| B.2 Examples of using <code>cilk_spawn</code> and <code>cilk_sync</code> | 333 |
| B.3 Serial reduction in C++ and equivalent Cilk Plus code | 336 |
| B.4 Serial reduction in C and equivalent Cilk Plus code | 338 |
| B.5 Example of using <code>__sec_reduce</code> to reduce over string concatenation | 342 |
| B.6 Defining an elemental function | 344 |
| B.7 Calling an elemental function from a vectorizable loop..... | 345 |
| C.1 Example of <code>affinity_partitioner</code> | 353 |
| C.2 Using <code>task_group</code> | 355 |
| C.3 Example of using <code>atomic<int></code> as a counter | 356 |
| C.4 Using atomic operations on a list..... | 357 |
| D.1 Using a manually written functor comparator | 362 |
| D.2 Using a lambda expression lets Listing D.1 be rewritten more concisely | 363 |
| D.3 Mixed capture with handwritten functor | 364 |
| D.4 Mixed capture modes..... | 364 |