OpenMP 4.1 and 5.0 Progress

Tuesday, November 18, 2014

Bronis R. de Supinski
Chair, OpenMP Language Committee

Lawrence Livermore National Laboratory

LLNL-PRES-664296

Plan for OpenMP specifications

- Recent Technical Reports
  - TR2 (March) details OpenMP performance tool support
  - TR3 (November) provides current snapshot of OpenMP 4.1
- OpenMP 4.1
  - Clarifications and errata to existing specification
  - Refinements and minor extensions
  - Do not break existing code
  - Minimal implementation burden beyond 4.0
  - Will be released by SC15
- OpenMP 5.0
  - Address several major open issues for OpenMP
    - Expect less significant advance than 4.0 from 3.1/3.0
  - Do not break existing code unnecessarily
  - Targeting release for SC17 (somewhat ambitious)
  - Plan to release intermediate TR(s) annually

TR3: An initial 4.1 comment draft

- Technical report adopted by ARB last week
  - Changes already approved by Language Committee
  - Includes 52 passed tickets (4.0 included 106 total)
  - Provide clear guidance to begin 4.1 implementations
- First released version following LaTeX conversion
  - Follows extensive proofreading passes
  - May include minor conversion errors
- Available for download at www.openmp.org
  - Full PDF version of initial 4.1 draft
    - Change history (Appendix E) is incomplete
  - A latexdiff from 4.0 is also available

TR3 already refines OpenMP device constructs significantly

- Added flush to several device constructs
- Unstructured data movement
- Require assignment for map (`always`)
- Improved asynchronous execution
  - Could have been in a task with just a `target` region
  - `target` and other device regions are now tasks
    - By default, undeferred
    - Can use `nowait` and `depend` clauses
- Many clarifications and minor corrections

Device constructs now support unstructured data movement

- Recast device data environment semantics
  - An implicit `target` region surrounds each initial thread
  - That region has ICVs are manipulated in the same ways as host ICVs
  - Variables are mapped into and out of that region's environment
- Use `target enter data` directive to map variables to devices

```
#pragma omp target enter data [clause [[,] clause] …]
```

  - Only `to` and `alloc` map types are allowed
- Use `target exit data` directive to unmap variables

```
#pragma omp target exit data [clause [[,] clause] …]
```

  - Only `from`, `release` and `delete` map types are allowed
- Semantics of `map` clauses recast as manipulating reference counts
- Use `map` clauses on `target` and `target data` directives to combine `enter` and `exit` semantics

TR3 adds new features to handle dependences within loops

- Added parameter to `ordered` clause
  - Specifies number of loops covered by `ordered` constructs
  - Similar to `collapse` clause
- Added clauses to `ordered` directive

```
#pragma omp ordered [clause [[,] clause] …]
```

  - Use `threads` with a structured data block for previous (default) semantics
  - Use `simd` with a structured data block to restrict to a single SIMD lane
  - Use `depend` without a structured data block for doacross semantics
- Doacross semantics restrict (but not prohibit) parallelism
  - Automates transformations such as loop skewing
  - Requires new dependence types:
    - `source` specifies iteration completes cross-ieteration dependences
    - `sink` with `vec` parameter blocks execution until specified `source` iterations complete

Can now easily specify that iterations of a loop should be divided into tasks

- Use `taskloop` directive to execute loop(s) as tasks

```
#pragma omp taskloop [clause [[,] clause] …]
```

  - Allows threads in team not to participate in loop execution
  - Supports concurrent asynchronous operations in non-loop tasks
- Partitions iterations of following loop into tasks
  - Avoids manual loop fission to aggregate iterations
  - Can use `grainsize` clause to specify iterations per task
    - Specifies a range of of iterations to support efficient task generation
  - Can use `num_tasks` clause to specify an exact number of tasks
- Accepts most clauses applicable to loop and `task` constructs
  - `if` generates undeferred tasks (i.e., it has tasking semantics)
  - `reduction` support not yet provided (may wait until OpenMP 5.0)

TR3 includes many other refinements to recent additions

- Many clarifications and minor enhancements
  - SIMD extensions
  - Thread affinity policies
  - Various other locations
- Reductions for C/C++ arrays
- Improved support for C++ reference types
- Reduced list of unsupported Fortran 2003 features by four bullets
- New terms to simplify discussion of new features

4.1 will refine OpenMP device constructs even further

- Additional clarifications (e.g., array section pointer)
- Refinements of combined clauses
  - Addition of even more combined constructs
  - Specifying overlapping clauses on combined constructs
- `memcpy` API to support manual mapping
- Interoperability with low-level device code (e.g., libraries)
- Tweaks to device environment support, including
  - `deferred_map` clause for `declare target` construct
  - Default ampping for scalar variables
  - Link clause/linkable support
- Deep copy/map/serialization for map
- Providing device-specific environment variables
- Multiple device types

Many other refinements to recent additions being considered for 4.1

- Still more clarifications and minor enhancements
  - SIMD and SIMD parallel loop chunk size control
  - Grammar for `OMP_PLACES`
- Continue to increase Fortran 2003 support
  - Ten limitations remain in TR3
  - Some limitations may remain until 5.0
- Initial support for memory affinity
- Interoperability with Pthreads
- Task priorities (almost done)
- Tasking and stand-alone reductions (5.0?)

More significant topics are being considered for OpenMP 5.0

- Initial support for recent C and C++ standards
- More tasking advances
  - Parallelism without `parallel` regions
  - Support for event loops
- Continued improvements to device support
- Performance and debugging tools support
- Interoperability and composability
- Locality and affinity
- General error model
- Transactional memory
- Additional looping constructs and refinements

Help us shape the future of OpenMP

- OpenMP continues to grow
  - 25 members currently
  - More expected in the near future
- You can contribute to our now planned annual TR or complete specification releases
- Attend IWOMP, become a cOMPunity member
- Changes to OpenMP membership types may soon support inexpensive academic memberships
  - Please let us know if you would be interested


Lawrence Livermore
National Laboratory