SAND2014-19699PE



Recent and Upcoming Enhancements to OpenMP

Stephen Olivier
Dept. 1423, Sandia National Laboratories, NM

Technical Seminar

Sandia Co-Design Working Group

November 12, 2014

Sandia National Laboratories

Exceptional service in the national interest

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline

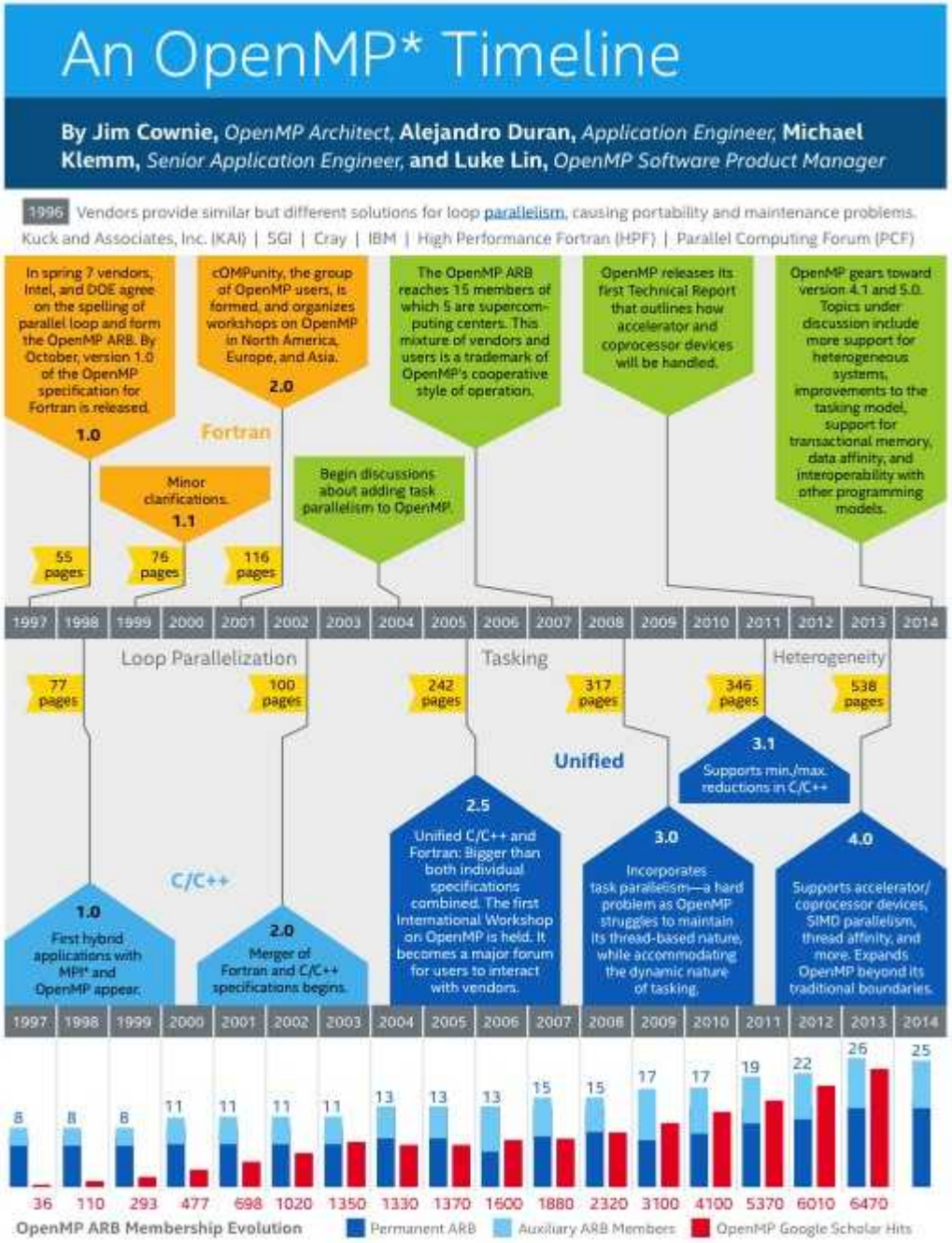
Sandia National Laboratories

- Brief history of OpenMP
- OpenMP 4.0 features
  - Focus on accelerator support
- Compiler support for 4.0
- Plans for 4.1 and beyond
  - Specification
  - OpenMP Tools Interface

Sandia National Laboratories

An OpenMP* Timeline

By Jim Cownie, *OpenMP Architect,* **Alejandro Duran,** *Application Engineer,* **Michael Klemm,** *Senior Application Engineer,* **and Luke Lin,** *OpenMP Software Product Manager*



Brief Summary of 4.0 Features

Sandia National Laboratories

- Accelerator device support
- SIMD (loop vectorization)
- Thread affinity
- Task dependences and task groups
- User-defined reductions
- Cancellation of parallel regions, loops, and task groups
- Some Fortran 2003 support
- Examples now in separate document

Portable OpenMP for GPU/Phi/CPU

Sandia National Laboratories

#pragma omp target teams distribute parallel for simd

- Yes, this is a lot to digest.

Portable OpenMP for GPU/Phi/CPU

Sandia National Laboratories

#pragma omp target teams distribute parallel for simd

Portable OpenMP for GPU/Phi/CPU

Sandia National Laboratories

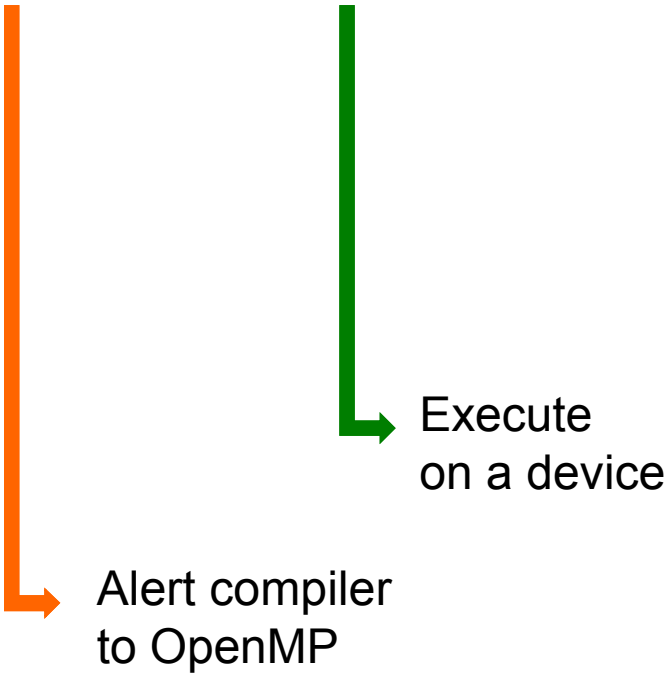
#pragma omp target teams distribute parallel for simd

Alert compiler
to OpenMP

Portable OpenMP for GPU/Phi/CPU

Sandia National Laboratories

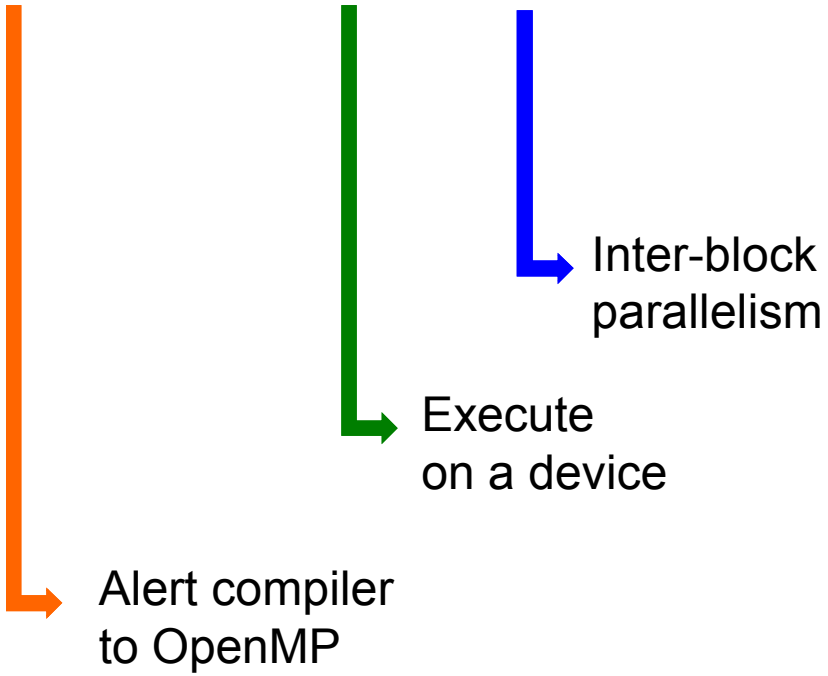
#pragma omp target teams distribute parallel for simd



Portable OpenMP for GPU/Phi/CPU

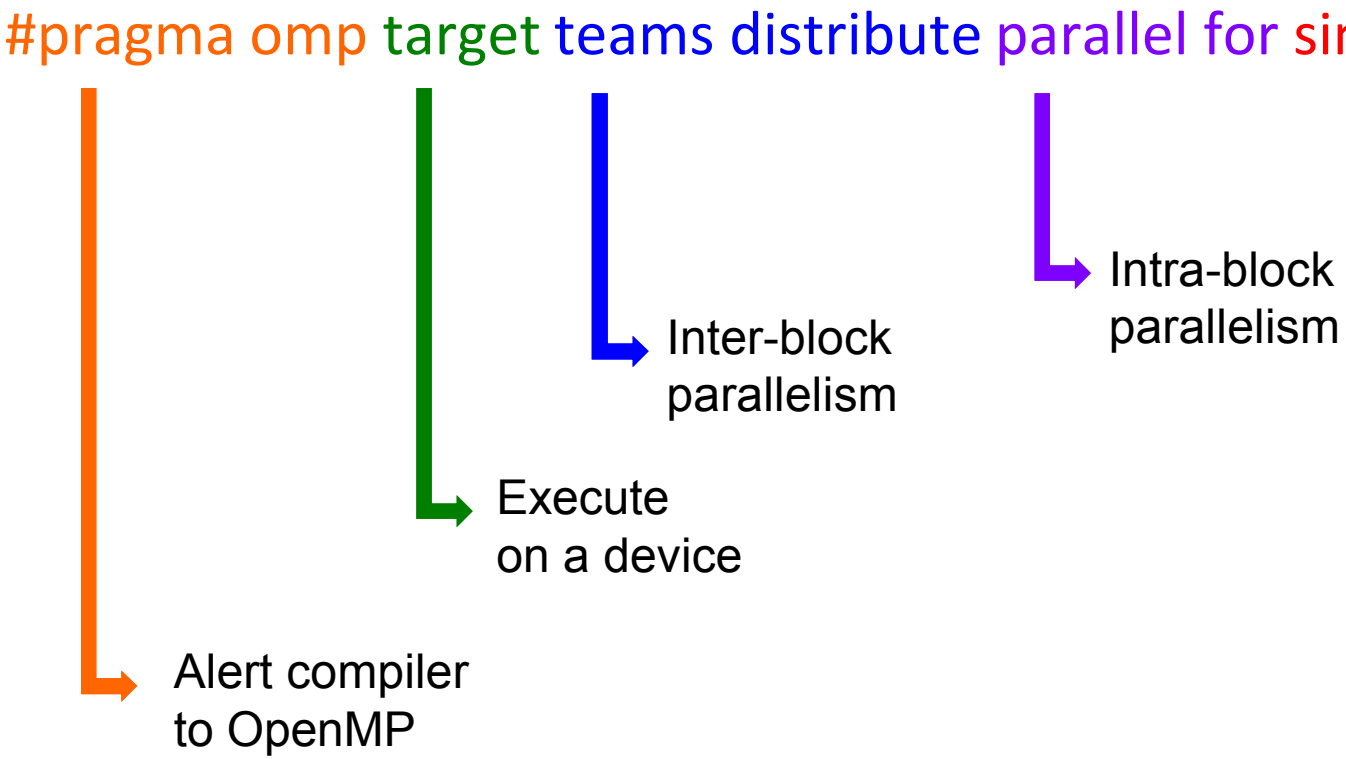
Sandia National Laboratories

```
#pragma omp target teams distribute parallel for simd
```



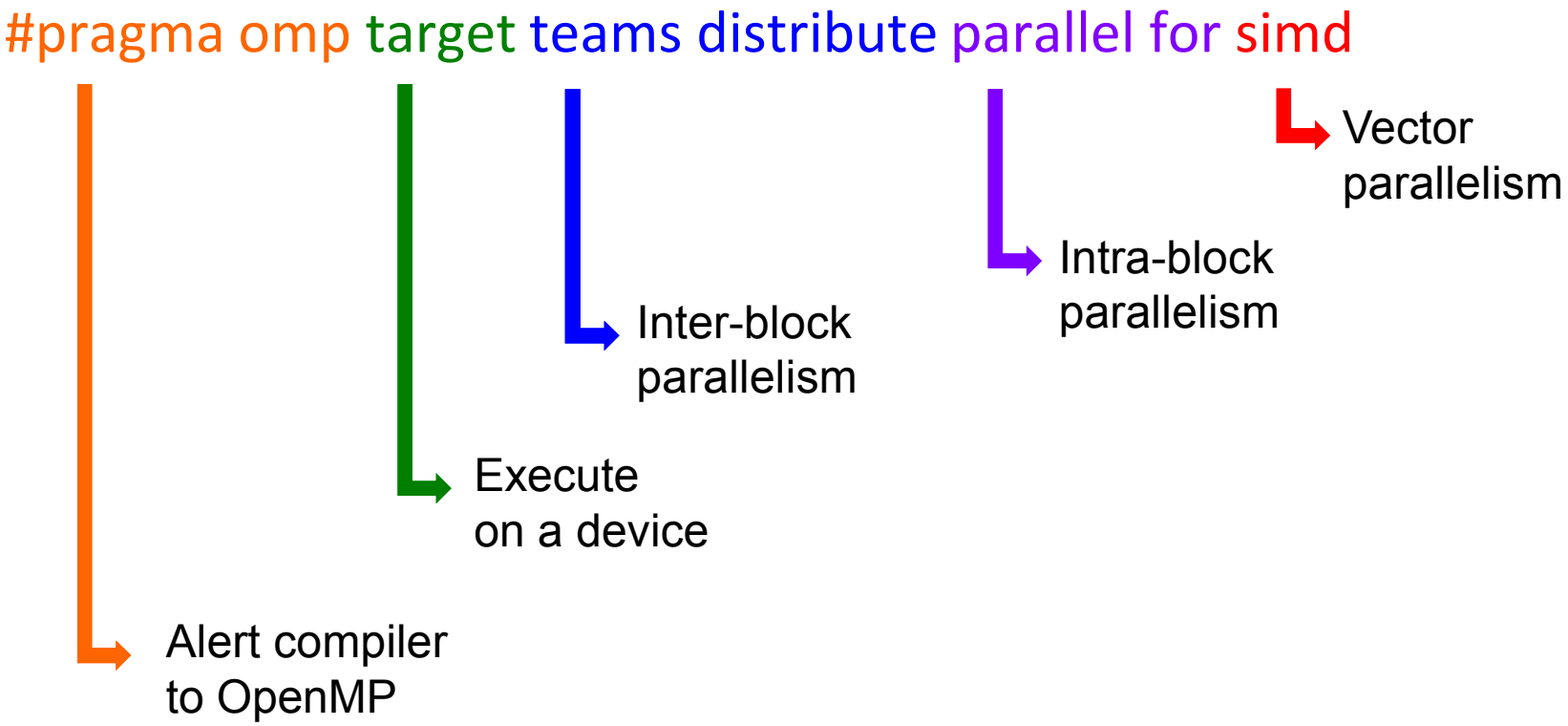
Portable OpenMP for GPU/Phi/CPU

Sandia National Laboratories



Portable OpenMP for GPU/Phi/CPU

Sandia National Laboratories



Resulting Device Parallelism

Sandia National Laboratories

- Inter-block (teams distribute)
  - Create a **league of thread teams** across the device (GPU or Phi)
  - Distribute chunks of loop iterations among the teams
  - Synchronization not allowed across teams
- Intra-block (parallel for)
  - **Each thread team** executes on a GPU block or Phi core(s)
  - Each thread executes sub-chunk(s) of its team's total chunk
  - Synchronization allowed within the team
- Vector-level (simd)
  - **Vectorize** for GPU warp or Phi AVX

Host/Device Data Mapping

Sandia National Laboratories

map(*map-type*: *list*)

- Clause on the **target** and **target data** constructs
- The *map-type* is one of
  - **to** (map to device on entry)
  - **from** (map from device on exit)
  - **tofrom** (map to device on entry and back from device on exit)
  - **alloc** (new uninitialized storage)
- The *list* specifies your variables to be moved
- Prefer to say “map” rather than “move”
  - Movement may not always be required

Updating data mid-region

Sandia National Laboratories

#pragma omp target update *motion-clause*

- Updates host / device data
  - Only other guaranteed updates are at start and finish of **target** and **target data** regions as specified in **map** clauses

- The *motion-clause* is one of
  - **to** (map to device on entry)
  - **from** (map from device on exit)

Example Code

Sandia National Laboratories

```
#pragma omp target teams distribute parallel for simd \
        map(to: x[0:N]) map(tofrom: y[0:N]) \
        num_teams(nblocks) \
        num_threads (nthreads)
 for (int i = 0; I < n; ++i) {
     y[i] = x[i] + y[i];
  }
```

Functions on device

Sandia National Laboratories

#pragma omp declare target

- Let compiler know that a function should exist on the device

Functions for SIMD

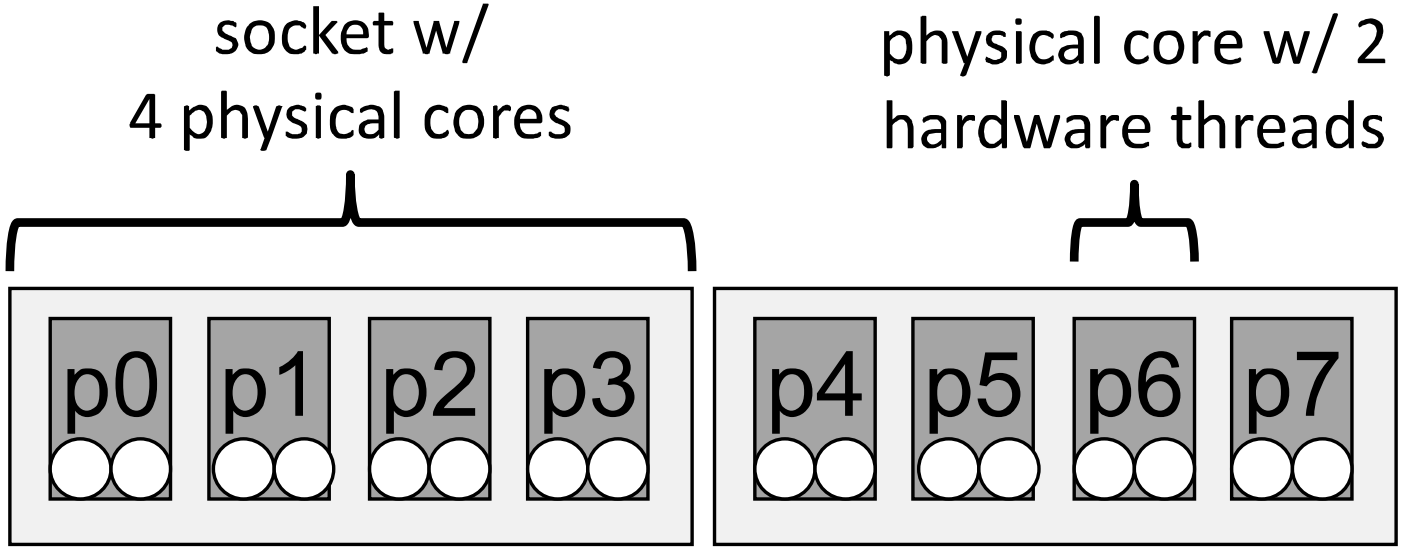
Sandia National Laboratories

#pragma omp declare simd

- Let compiler know that a function can be used inside a vectorizable loop

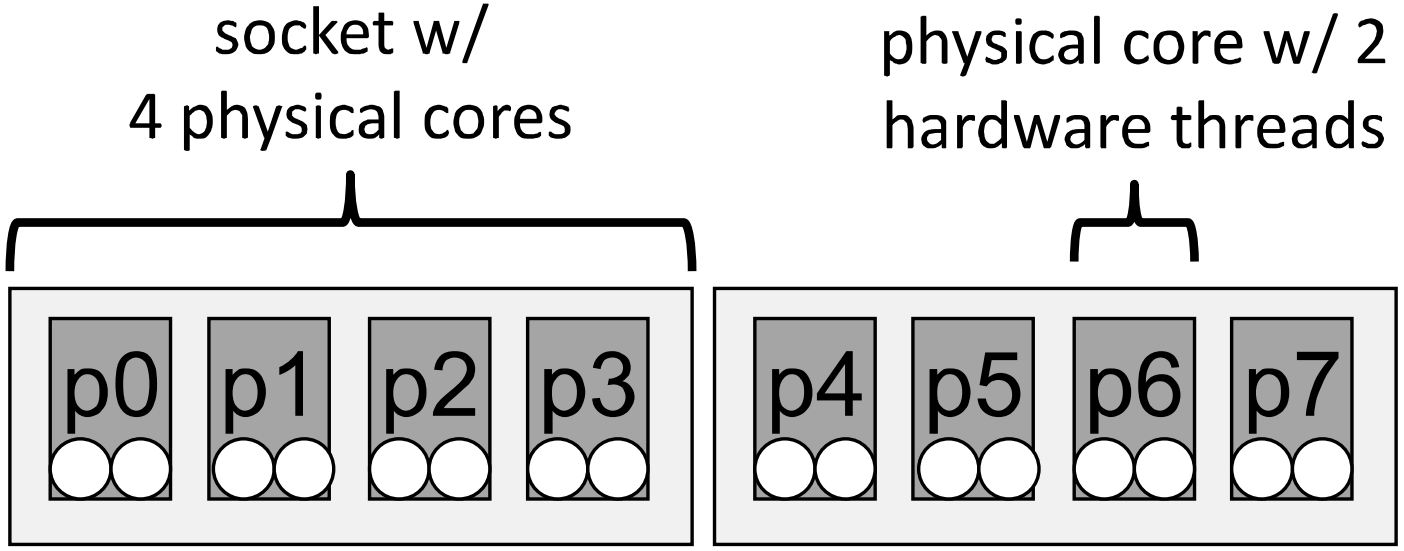
- New clauses to specify
  - Alignment
  - Used in branching code or not
  - Safe vector length

Thread Affinity (Example)



- Define a place list grouping by core:
  - OMP_PLACES="{0,1},{2,3},{4,5},{6,7},{8,9},{10,11},{12,13},{14,15}"
  - Or OMP_PLACES=cores
- Or by socket:
  - OMP_PLACES="{0,1,2,3,4,5,6,7},{8,9,10,11,12,13,14,15}"
  - Or OMP_PLACES=sockets

Thread Affinity (Example)



- Suppose we specify
  - OMP_PLACES="{0,1},{2,3},{4,5},{6,7},{8,9},{10,11},{12,13},{14,15}"
- And we want one thread per core
  - PROC_BIND="spread" NUM_THREADS=8
- Can use PROC_BIND="close" to put threads close together

Task Dependences

Sandia National Laboratories

- Express relationships between tasks based on data flow
  - Flow dependence (read-after-write)
  - Anti-dependence (write-after-read)
  - Output dependence (write-after-write)
- Express using **in** and **out** dependence clause lists
- Can use dummy variables to create arbitrary dependence graphs as needed

Task Dependences (Example)

Sandia National Laboratories

```
int x = 1;
#pragma omp parallel
#pragma omp single
{
        #pragma omp task shared(x)
                x = 2;
        #pragma omp task shared(x)
                printf ("x = %d\n", x);
}

// Could print “x = 1” or “x = 2”
```

Task Dependences (Example)

Sandia National Laboratories

```
int x = 1;
#pragma omp parallel
#pragma omp single
{
        #pragma omp task shared(x) depend(out: x)
                x = 2;
        #pragma omp task shared(x) depend(in: x)
                printf ("x = %d\n", x);
}

// Will always print “x = 2”
```

User-defined Reductions

Sandia National Laboratories

- Previously, reductions defined over particular set of operators
  - Mathematical operators **(+**, **-**, *, **&**,**|**,^,**&&,**and**||)**
  - OpenMP 3.1 added **min** and **max**

- Now can define your own reductions
  - Specify initializer and combiner functions

Cancellation

Sandia National Laboratories

- Can prematurely stop execution
  - Defined for Parallel regions, loops, task groups
- Support error conditions or terminating search when the answer is found
- Some default cancellation points
  - Others can be marked by user
- Part of a larger effort to define an error model for OpenMP
  - (Currently none)

Compilers Supporting OpenMP 4.0

Sandia National Laboratories

- GCC 4.9
  - Device constructs will execute on host
- Intel 14.0
  - Except user-defined reductions
- Clang/LLVM
  - Not in trunk; available at http://clang-omp.github.io/
  - Based on Intel open-source RTL
- Oracle 12.4
  - Except SIMD and device constructs

Process for Future of OpenMP

Sandia National Laboratories

- OpenMP 4.1 to be released at SC15
  - SC14 tech report shows progress so far
- OpenMP 5.0 tentatively released at SC17
- Separate examples document
  - Allows updates off-cycle with new spec. versions
  - Moving toward whole compilable, runnable programs

On deck for OpenMP 4.1

Sandia National Laboratories

- Asynchronous accelerator execution
- Unstructured data mapping
- Array reductions
- DOACROSS (structured loop dependences)
- Task-generating loops
- More Fortran 2003
- Many corrections

Tech Report with preliminary 4.1 features to be released at SC14

OpenMP Tools (OMPT) Interface

Sandia National Laboratories

- Released as tech report in March 2014
  - http://openmp.org/mp-documents/ompt-tr2.pdf
  - Not required of implementations to be compliant with OpenMP spec.
- Designed for minimal run time system overhead
- Tracks threads, parallel regions, and threads
  - Unique identifiers for each
  - Callbacks at begin and end of each
  - Record thread states (idle, serial, parallel)
- Tracks waiting on barriers, locks, critical sections, child tasks
- Many optional events for instrumentation

OpenMP 5.0 and Beyond

Sandia National Laboratories

- Multiple accelerator device types
- Memory placement / affinity
- Task-to-thread mapping and task reductions
- Interoperability / composability
- Tools support (in spec)
- Transactional memory

OpenMP Resources

Sandia National Laboratories

- Main site, including the spec and examples document:
  - http://www.openmp.org
- Excellent article on OpenMP 4.0 from Intel:
  - https://software.intel.com/sites/default/files/managed/64/cc/parallel_mag_issue16.pdf
- Book (unfortunately not updated for recent OpenMP spec.):
  - http://mitpress.mit.edu/books/using-openmp
- SC BoF on Tuesday 5:30-7pm
- SC OpenMP booth (#2824)
  - Booth talks Tuesday – Thursday
  - Free snacks Tus