

# Status of OpenMP 3.1 and 4.0 Specifications

## June 16, 2010



**Bronis R. de Supinski**  
**OpenMP Language Committee Chair**  
**Center for Applied Scientific Computing**  
**Science & Technology Principal Directorate - Computation Directorate**

# We are in a time of great opportunity and challenge for the OpenMP programming model



- Multicore systems are prevalent, will dominate the near term
  - Naturally suited to thread-based parallel programming
  - Decreasing memory per core
  - Decreasing message rates per core
- Heterogeneous systems exist and may dominate long term
  - Suitability of OpenMP model?
  - Some think OpenCL is enough...
- Many make claims about OpenMP that are inaccurate:
  - “The model does not scale.”
  - “Other models better capture machine structure.”
  - “MPI is here to stay.”
  - “Hybrid programming models are too complex.”



# We have well defined plans for the next two OpenMP specifications



- OpenMP 3.1
  - Refine and extend existing specification
  - Do not break existing code
  - Minimal implementation burden beyond 3.0
  - Draft very soon after IWOMP 2010
    - Slightly delayed from previously announced schedule
  - Released specification approximately SC 2010
- OpenMP 4.0
  - Address several major open issues for OpenMP
  - Do not break existing code unnecessarily
  - Draft planned/hoped for SC 2011



# The OpenMP Language Committee uses an open and democratic process



- Voting rights by organization
  - Attendance requirements
  - Contention-free issues handled by simple voice vote
- Regular meetings
  - Weekly teleconferences
  - Tri-annual face-to-face meetings (next: later this week)
- Instituted issue tracking mechanism for 3.1
  - Tickets include description of solution and alternatives
  - Provides easy mechanism for distributed work
- Specification currently contained in FrameMaker document
  - Richard Friedman serving as 3.1 editor
  - Planning conversion to LaTex for 4.0 (Thanks, Alex!)



# Despite incremental nature, we are targeting several important items for OpenMP 3.1



- Add user-defined reduction operators
  - Proposed support discussed in Alejandro Duran's talk
  - Some issues created contention
    - Array reduction support in C/C++
    - Template support
  - To be resolved in Language Committee F2F this week
- Extend atomics to support capture and write functionality
- Small extensions to OpenMP tasking model
  - Explicit task scheduling points (**taskyield** construct)
  - Ability to save data environment overhead (**final** clause)
- Many clarifications, including improvements to examples



# Additional kind of atomic operations addresses an obvious deficiency



- Currently cannot capture a value atomically

```
int schedule (int upper) {  
    static int iter = 0; int ret;  
    ret = iter;  
    #pragma omp atomic  
        iter++;  
    if (ret <= upper) { return ret; }  
    else { return -1; } //no more iters  
}
```

- Atomic capture provides the needed functionality

```
int schedule (int upper) {  
    static int iter = 0; int ret;  
    #pragma omp atomic capture  
        ret = iter++; // atomic capture  
    if (ret <= upper) { return ret; }  
    else { return -1; } // no more iters  
}
```



# We are actively discussing several major topics for OpenMP 4.0



- Development of an error model -- Michael Wong's talk
  - The done directive
  - Functionality like errno
  - Callbacks for integrated error handling
- Interoperability and composability
  - Interactions between thread models
  - Interoperability of OpenMP implementations
- Processor and data affinity
  - How to specify subarrays in C
  - Broader discussions about “place” also
- Refinements to the OpenMP tasking model



## 4.0 will at least add high-level affinity support

- Control of nested thread team sizes

```
export OMP_NUM_THREADS=4,3,2
```

- Instruct run time to bind threads to resources

```
export OMP_PROC_BIND=TRUE
```

- Restrict the processor set for program execution

```
export OMP_PROCSET 0,1,2,3,8,10,12,14
```

- Control thread placement within a processor set

```
export OMP_AFFINITY=scatter,scatter,compact
```

- Control initial placement of shared data

```
export OMP_MEMORY_PLACEMENT=spread | local
```

- Adapt data placement at runtime

```
#pragma omp migrate (list) strategy (local | spread)
```



# Support for accelerators is a major new direction for OpenMP 4.0



- Initial work based on prototypes from several vendors
- Current expectation is to use an accelerator tasking model
  - Include a clause to indicate accelerator region
  - Generate an explicit task
    - Probably tied; limited to same accelerator type if untied
    - Run to completion
    - Complete at barriers, accelerator task sync points
  - Explicit device teams, scheduled by run time
- Discussing a wide range of issues
  - Memory model including data environment changes
  - Implications for loop construct
  - Possible restrictions of other constructs



# Several Other Topics under Consideration for OpenMP 4.0



- Other topics being considered for OpenMP 4.0
  - Transactional memory and thread level speculation
  - Additional task/thread synchronization mechanisms
  - Extending OpenMP to Fortran 2003
  - Extending OpenMP to additional languages
  - Incorporating tools support
  - Other miscellaneous extensions
- How can you participate in OpenMP?
  - Attend IWOMP, become a cOMPunity member
  - Lobby your institution to join the OpenMP ARB
  - Contact me and beg ;-)

