Summary

The OpenMP BoF was held on Tuesday 15 November 5:15-7:00pm.

There were >120 attendees.

Oral feedback was positive, feedback from the SC16 web-based tools was also good (though since only two attendees provided it the best we can likely conclude is that people weren’t so annoyed they felt the need to down-vote it!)

A large component of the BoF involved audience questions and discussion, which grew heated in a few places. (I had to truncate a discussion between a vendor and one of their customers, since it had strayed from OpenMP to more general issues of compiler delivery and support!)

I believe the BoF achieved the aims of a BoF

- Education
- Community building
- Audience interaction (>70% of the time was devoted to discussion)
Slide 1

**OpenMP® Birds of a Feather**

Programme:

- OpenMP TR4 introduction (5m)  
  Bronis R. de Supinski  
  Chair of Language Committee
- Device Constructs Update (3m)  
  Eric Stozer  
  Texas Instruments
- Affinity & Memory (3m)  
  Christian Terboven  
  RWTH Aachen
- SIMD Extensions (3m)  
  Xinmin Tian  
  Intel
- Tools Support (3m)  
  John Mellor-Crummey  
  Rice University
- Tasks (3m)  
  Stephen Olivier  
  Sandia
- Quiz the TR4 experts (30-40m, until you run out of questions!)
- OpenMP ARB (5m)  
  Michael Klemm  
  OpenMP CEO
- Quiz the ARB (40m or so)

Slide 2

**TR4 and Language Committee Overview**

Tuesday, November 15, 2016  
Bronis R. de Supinski  
Chair, OpenMP Language Committee
TR4 released November 2016

- Includes 24 passed tickets
  - 72 other tickets in progress, other topics not yet in tickets
- Major new feature is performance tool support (TR2+)
- Some significant extensions to existing functionality
  - Support for task reductions, including on taskloop construct
  - Implicit declare target directives and other verbosity reducing changes
- Many clarifications and minor enhancements, including:
  - Use of any C/C++ lvalue in depend clauses
  - Addition of depend clause to taskwait construct
  - Addition of conditional modifier to lastprivate clause
  - Permits declare target on C++ classes with virtual members
  - Clarification of declare target C++ initializations

OpenMP 5.0 will significantly extend TR4

- OpenMP 5.0 will be released by SC18
  - Plan to release TR5 on support for heterogeneous memories soon
  - Another TR (TR6?) next November will document most additions for 5.0
- Topics for 5.0 beyond TR4
  - Memory locality, affinity and working with complex memory hierarchies
  - Updates to support latest C/C++ standards, completion of Fortran 2003
  - Continued improvements to device support and tasking, including:
    - Deep copy for mapped variables; Improved support for multiple devices
    - Unshackled threads, major extensions for task dependences
  - Interoperability and composability
  - Debugging tools support
Help us shape the future of OpenMP

- OpenMP continues to grow
  - 28 members currently
- You can contribute to our now planned annual TR or complete specification releases
- Attend IWOMP, become a cOMPunity member
- Changes to OpenMP membership types will (I think) soon support less expensive memberships
  - Please let us know if you would be interested
Device Constructs Update

OpenMP BOF SC16
Eric Stotzer

OpenMP TR5 Summary

• Clarifications
  – How variables are mapped
  – C/C++ initialization of declare target variables
  – Reduction clause on target construct

• Ease of Use
  – Allow static and virtual members in mappable class type
  – Implicit declare target for functions/routines
  – Partial mapping of structures
**OpenMP TR5 – implicit declare target**

- Make it easier to accelerate code with many function calls.
- Implicit `declare target` for functions/routines referenced in a target region.
- If the definition is in another file (see G()) it must be explicit or implicit `declare target` in that file.

```c
// This code fails with 4.5 but will work in TR5.
extern void G();
void bar()
{
  G(); // implicit declare target
}
void foo()
{
  #pragma omp target
  bar(); // implicit declare target
}
#pragma omp declare target to(G)
void G()
{
  // function definition
}
```

**OpenMP 5.0**

- Improve C++ and Fortran support
  - Virtual member functions
  - `declare target` for function pointers
- Deep Copy
  - Mapping a pointer and pointed-to memory
  - Mapping complex and unstructured data
- Multiple device types
  - Generalize to multiple types of devices
  - Customize code for specific device types
Affinity

Memory Management

Christian Terboven

The Status of Affinity in OpenMP

- We worked towards the support of Task Affinity:
  - Affinity of task to data, or data to tasks, or both?
  - And we realized we would like to have support for memory management first!
Slide 13

Let me demonstrate with an example!

- Other memspace: high bandwidth memory
  ```c
  omp_memtrait_set_t trait_set;
  omp_init_memtrait_set(&trait_set, 1,
    {{OMP_MTK_OPTIMIZED, OMP_MTK_BANDWIDTH}});
  ```

- Memspace: memory resource in the system
  ```c
  omp_memspace_t *mspace =
  omp_init_memspace(omp_default_memtraits);
  ```

- Allocator: object to perform memory management
  ```c
 omp_allocator_t *allocator =
  omp_init_allocator(mspace, omp_default_alloctraits);
  ```

- Separation of property selection and allocation
  ```c
double *buf = omp_alloc(n * sizeof(*buf), allocator);
or
#pragma omp allocates(var) allocator(...) memspace(...)
  ```

Slide 14

Summary and Next Steps

- Memory Management builds the foundation for
  - Exploitation of multiple memory kinds
  - Exploitation of NUMA
  - High-level abstractions, like Task Affinity
  - ...

- Next steps:
  - Definition of traits, complete integration with OpenMP
  - Simplification?!

- New concept in OpenMP. We want feedback.
  - Release of a TR document planned for early 2017
**SIMD Extensions for Effectiveness**

- Added "simd" clause to "ordered" construct
- New modifiers (ref | uval | val) for "linear" clause

```cpp
#pragma omp simd
for (i = 0; i < N; i++) {
    ...
#pragma omp ordered simd
    { if (b[i]) > 0) q[j++] = b[i];
    }
    ...
}
#pragma omp declare simd notinbranch \ linear(ref(p))
int add_one(const int *p) {
    return (p + 1);
}
```

```asm
2 instructions vs. 15 instructions
```

---

**SC'2016 OpenMP* BoF: SIMD Extensions and Beyond**

Xinmin Tian, Senior Principal Engineer
Intel Corporation
Future SIMD Extensions

- Declare SIMD for function pointers / virtual functions / lambda function
  - Work in progress for 5.0 beta
- Compress/expand/conflict/histogram support
  - Work in progress for 5.0 beta
- Allow SIMD for C++ range-based for loops
  - Work in progress for 5.0 beta
- Conditional lastprivate
  - Voted in for 5.0 Alpha (TR4)

SIMD extensions are semantics oriented and architecture agnostic

OMPT: A Tools Interface for OpenMP

- Goal: Enable development of high-quality, first-party tools that support monitoring and performance analysis of OpenMP programs
  - Enable tools to work with any OpenMP implementation on any platform
  - Enable tools to map implementation-level performance measurements back to source-level abstractions
- Design based on experiences with prototypes in several runtimes and tools
- OMPT tools interface integrated into OpenMP v5.0 preview 1 (bit.ly/openmp-tr4)
  - Tool initialization and callback registration
  - Monitoring OpenMP on host
    - Callback interfaces that notify tools about execution of OpenMP constructs and runtime routines
    - Entry points that tools use for runtime introspection: states, parallel regions, tasking
  - Monitoring OpenMP on target devices
    - Trace-based design
    - Prototypes for target devices including Xeon Phi, NVIDIA GPU, FPGA
- IBM is already working on a full implementation of OMPT in LOMP
- Implementations in other runtimes expected to begin soon

Contacts: John Mellor-Crummey (johnmc@rice.edu), or OMP Tools Group (omp-tools@openmp.org)
OMPD: A Debugging Interface for OpenMP

- Goal: Enable development of high-quality, third-party debuggers that work with any OpenMP implementation on any platform
- Specification is publicly available (bit.ly/ompd-tr)
  - Last refined in July 2016
- OMPD prototype for OpenMP 3 is almost ready for release in Clang runtime
  - Currently maintaining code in a private repo at LLNL
  - Expecting a release in early 2017
  - Developing standalone tests to include in Clang runtime
- Prototype is being tested/developed using multiple debugging frameworks
  - TotalView, gdb, DynInst
- IBM has begun developing OMPD support in LOMP
- Work has begun on OMPD for OpenMP 4/4.5 target constructs
  - First focus: NVIDIA GPU accelerators
  - Prototype planned for Apr/May, 2017

Contacts: Ignacio Laguna (ilaguna@llnl.gov), or OMP Tools Group (omp-tools@openmp.org)
Task Reductions (Finally!)

- To get something like a reduction among tasks before TR4:
  - Option 1: Collect partial values on each thread in threadprivate variables
  - Option 2: Combine and pass partial values through tree of tasks
  - Option 3: Locks / atomic / critical to protect “reduction variable”

- New in TR4: Reductions with taskgroup or taskloop
  - Reduction over a task group:
    - Use task_reduction clause on the taskgroup construct
    - Use in_reduction clause on tasks within the task group
  - Reductions with the taskloop construct:
    - Use reduction clause for new reduction over its generated tasks
    - Use in_reduction clause to participate in a previously defined reduction
Task Reductions Example

```c
list_item_t list_item = list->head;
int sum = 0;
#pragma omp taskgroup task_reduction (+:sum)
{
    while (list_item) {
       膘#pragma omp task in_reduction (+:sum)
        {
            sum += visit(list_item);
            list_item = list_item->next;
        }
    }
}
```

Other Enhancements

- Using the depend clause on the taskwait construct
  - **Before TR4, taskwait takes no clauses**
  - Synchronize all child tasks created to that point
  - **New in TR4, taskwait allows the depend clause**
  - Synchronize child tasks based on the depend clause semantics
  - Like an empty merged task with a depend clause
- Expand variable types allowed in the depend clause
  - **New in TR4, any C/C++ l-value may be used**
- Clarifications of tasking concepts and terminology
  - Nesting of regions
  - Task-generating constructs
Ask the TR4 experts

Questions

OpenMP ARB
Michael Klemm
michael.klemm@openmp.org
Slide 27

OpenMP ARB Mission

The mission of the OpenMP ARB (Architecture Review Board) is to standardize directive-based multi-language high-level parallelism that is performant, productive and portable.

Slide 28

OpenMP ARB

- OpenMP membership continues to grow
  - Currently 29 members (13 permanent and 16 auxiliary)
  - We are actively looking for new members
- The ARB is working on changes to OpenMP membership categories
- How you can get involved:
  - Attend OpenMPCon, IWOMP
  - Become a cOMPunity member
Planned New Membership Structure

- **ARB Member**
  - Highest membership category
  - Participation in technical discussions and organizational decisions
  - Voting rights on technical topics (tickets, TRs, specifications)
  - Voting rights on organizational topics

- **ARB Advisor & ARB Contributors**
  - Contributors to technical discussions
  - Voting rights on technical topics (tickets, TRs, specifications)
ARB Question Session

ARB Panel:
Alice Koniges Lawrence Berkeley National Lab
Barbara Chapman Stony Brook University (& cOMPunity)
Bronis R. de Supinski Chair of OpenMP Language Committee
Doug Miles NVIDIA (Portland Group)
Kelvin Li IBM
Luiz de Rose Cray
Michael Klemm OpenMP CEO
Simon McIntosh-Smith University of Bristol
Xinmin Tian Intel