OMPT Breakout Report

Local:
Andreas Knuepfer (TU Dresden), Leonardo Fialho (TACC), Kevin Huck (Oregon), John Mellor-Crummey (Rice), Martin Schulz (LLNL),

Remote:
Tim Cramer (RWTH Aachen), John Del Signore (Rogue Wave), Alexandre Eichenberger (IBM), Ignacio Laguna (LLNL), Daniel Lorenz (TU Darmstadt), Joachim Protze (RWTH Aachen), Mark Schlueter (JSC)

Agenda

Review several aspects of OMPT design

- Task Dependence Tracking
- TARGET callback events
- TARGET device tracing
- TARGET inquiry functions

Task Dependence Tracking

- Report task dependences with a separate callback rather than as part of explicit task creation
  - avoid having assembling information that a tool may not want
- Report task dependences on variables and in/out/inout
  - replace version that reports dependences between task pairs

```
/* task dependences */
typedef void (*ompt_task_dependence_callback_t) (
  ompt_task_id_t task_id,       /* ID of task      */
  ompt_task_dependence_t *dependence_vector,
  int vector_length
);

typedef struct {
     void *              base_addr;
     size_t              len;
     struct { bool   in:1; bool  out:1; } flags;
} ompt_task_dependence_t
```

TARGET Device Tracing

- Goal: allow OMPT tools to gather and report information reported in a native event trace without full knowledge of a target device
- Motivation: NVIDIA's rich CUPTI activity API can report many events
- New design

```
ompt_native_summary_t *ompt_get_record_native_summary(
   void *native_record
)

typedef struct {
     ompt_native_kind_t kind // info or event
       const char *type
       uint64_t start_time // -1 if not available
       uint64_t end_time // -1 if not available
       int hwid // -1 if not available
} opt_native_summary_t
```

Next Steps

- Revise document and circulate to tools committee
- Committee to review coverage of OpenMP Standard
  - missing support for CANCEL
- Begin process of offering OMPT to standards committee

Details Follow

TARGET Callback Events

- Revise document to indicate that TARGET data and TARGET map “end” event callbacks occur when execution finishes “on the host”
  - asynchronous execution on a device must be allowed to continue after the host signals the end event
- Replace “data_map_id” in TARGET map callback with pointers to host and device addresses
  - advantages
    - no management of map ids
    - useful for correctness checking
- Add new optional event “ompt_data_map_finished” to indicate end of asynchronous map operation

Associating TARGET Callbacks with Code

- Current design for TARGET callbacks (target, target data, target data map) reports pointer to outlined function
  - may not exist in some implementations
- Revised design: return an opaque handle for code
  - may represent an object that contains device code

TARGET Inquiry Functions

- Revisit design for obtaining device timestamps for synchronizing host and device clocks
  - check what CUPTI provides to make sure design is practical