Performance Analysis for Target Devices with the OpenMP Tools Interface

Tim Cramer, Robert Dietrich, Christian Terboven, Matthias S. Müller, Wolfgang E. Nagel
IT Center, RWTH Aachen University
Technische Universität Dresden

HIPS Workshop, Hyderabad, 25.05.2015
Motivation

- **Requirement for large compute capabilities**
  - Wide use and acceptance of new architectures
  - New user friendly programming paradigms (e.g., OpenMP 4.0, OpenACC)
  - But: adequate tools support is important as well

- **Tools support for OpenMP**
  - OpenMP Architecture Review Board (ARB) published a technical report describing the OpenMP Tools (OMPT) interface to extend the standard
  - Support only for OpenMP 3.1
Agenda

- **Introduction**
  - The OpenMP Tools Interface (OMPT)
  - The OpenMP 4.0 Target Constructs

- **Contributions**
  - Extension of OMPT for Target Constructs
  - Runtime Implementation (LLVM/Intel OpenMP)
  - Tool Integration (Measurements with Score-P, Visualization with Vampir)
  - Performance Evaluation

- **Conclusion**
OpenMP Tools Interface (OMPT)

- **OMPT Features**
  - Support for asynchronous sampling
  - Blame shifting techniques
  - Callbacks definitions for instrumentation-based monitoring of runtime events
  - Standardized interface

- **OMPT Design Objectives**
  - Should provide sufficient information about the program and the OpenMP runtime system
  - Low overhead API
  - Low development burden for the runtime and tool developer
  - OpenMP runtime
    - maintains information about the state of each thread
    - provides API calls to interrogate the runtime
OMPT: Benefits

**Benefits**

- Event-based performance analysis with low overhead
- Standardized interface (OMPT only for 3.1 at the moment)
- Runtime information (OMPT) vs. source-to-source instrumentation (OPARI2)
  - **OPARI2**
    - ☑️ can deliver exact user code mapping
    - ☑️ independent of OpenMP implementation
    - 😞 recompiling necessary
    - 😞 Low detail level of information, e.g.
      - data transfer size
      - mapping of variable or array
    - 😞 no chance for standardization
  - **OMPT is vice versa**
- **Overhead of both is similar** [1]

---

**Reference**

OpenMP 4.0: Target Constructs (SAXPY Example)

```c
int n = 10240; float a = 42.0f; float b = 23.0f;
float *x, *y;
// Allocate and initialize x, y
// Run SAXPY TWICE and process data on host
{
    #pragma omp parallel for
    for (int i = 0; i < n; ++i){
        y[i] = a*x[i] + y[i];
    }

    processDataOnHost(y);

    #pragma omp parallel for
    for (int i = 0; i < n; ++i){
        y[i] = b*x[i] + y[i];
    }
}
```

Host Device

```c
main()
{
    saxpy();
    processDataOnHost();
    saxpy();
}
```

Target Device
Performance Analysis for Target Devices with the OpenMP Tools Interface

Tim Cramer | IT Center der RWTH Aachen University

OpenMP 4.0: Target Constructs (SAXPY Example)

```c
int n = 10240; float a = 42.0f; float b = 23.0f;
float *x, *y;
// Allocate and initialize x, y
// Run SAXPY TWICE and process data on host
{
    #pragma omp target
    #pragma omp parallel for
    for (int i = 0; i < n; ++i){
        y[i] = a*x[i] + y[i];
    }

    processDataOnHost(y);

    #pragma omp target
    #pragma omp parallel for
    for (int i = 0; i < n; ++i){
        y[i] = b*x[i] + y[i];
    }
}
```

Host Device
```
main(){
saxpy();
processDataOnHost();
saxpy();
}
```

Target Device
```
saxpy();
saxpy();
```
int n = 10240; float a = 42.0f; float b = 23.0f;
float *x, *y;
// Allocate and initialize x, y
// Run SAXPY TWICE and process data on host
{
  #pragma omp target map(to:x[0:n]) map(tofrom:y[0:n])
  #pragma omp parallel for
  for (int i = 0; i < n; ++i){
    y[i] = a*x[i] + y[i];
  }
  processDataOnHost(y);

  #pragma omp target map(to:x[0:n]) map(tofrom:y[0:n])
  #pragma omp parallel for
  for (int i = 0; i < n; ++i){
    y[i] = b*x[i] + y[i];
  }
}

main(){
saxpy();
processDataOnHost();
saxpy();
}
OpenMP 4.0: Target Constructs (SAXPY Example)

```c
int n = 10240; float a = 42.0f; float b = 23.0f;
float *x, *y;
// Allocate and initialize x, y
// Run SAXPY TWICE and process data on host
#pragma omp target data map(to:x[0:n]) map(tofrom:y[0:n])
{
    #pragma omp target
    #pragma omp parallel for
    for (int i = 0; i < n; ++i){
        y[i] = a*x[i] + y[i];
    }

    processDataOnHost(y);

    #pragma omp target
    #pragma omp parallel for
    for (int i = 0; i < n; ++i){
        y[i] = b*x[i] + y[i];
    }
}
```

Host Device

```c
main(){
saxpy();
processDataOnHost();
saxpy();
}
```

Target Device

```c
saxpy();
saxpy();
```
OpenMP 4.0: Target Constructs (SAXPY Example)

```c
int n = 10240; float a = 42.0f; float b = 23.0f;
float *x, *y;
// Allocate and initialize x, y
// Run SAXPY TWICE and process data on host
#pragma omp target data map(to:x[0:n]) map(tofrom:y[0:n])
{
    #pragma omp target
    #pragma omp parallel for
    for (int i = 0; i < n; ++i){
        y[i] = a*x[i] + y[i];
    }
    #pragma omp target update from(y[0:n])
    processDataOnHost(y);
    #pragma omp target update to(y[0:n])
    #pragma omp target
    #pragma omp parallel for
    for (int i = 0; i < n; ++i){
        y[i] = b*x[i] + y[i];
    }
}
```

Performance Analysis for Target Devices with the OpenMP Tools Interface
Tim Cramer | IT Center der RWTH Aachen University
### Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification of constructs</td>
<td>Event (begin)</td>
<td>after</td>
</tr>
<tr>
<td>Host to device data map</td>
<td>Event (begin)</td>
<td>before</td>
</tr>
</tbody>
</table>

**OMPT: Begin Events for Target Directives**

**Performance Analysis for Target Devices with the OpenMP Tools Interface**

**Tim Cramer** | IT Center der RWTH Aachen University
### Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification of constructs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification of constructs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### OMPT: Begin Events for Target Directives

#### Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification of constructs</td>
<td>target</td>
<td>after</td>
</tr>
<tr>
<td></td>
<td>target data</td>
<td>before</td>
</tr>
<tr>
<td></td>
<td>target update</td>
<td></td>
</tr>
</tbody>
</table>
OMPT: Begin Events for Target Directives

Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>after</td>
</tr>
<tr>
<td>Identification of constructs</td>
<td>target</td>
<td>a task encounters a target construct</td>
</tr>
<tr>
<td></td>
<td>target data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>target update</td>
<td></td>
</tr>
</tbody>
</table>
## OMPT: Begin Events for Target Directives

### Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>constructs</td>
<td>target</td>
<td>after a task encounters</td>
</tr>
<tr>
<td></td>
<td>target data</td>
<td>a task encounters</td>
</tr>
<tr>
<td></td>
<td>target update</td>
<td>region is executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>new data environment is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>created</td>
</tr>
</tbody>
</table>
OMPT: Begin Events for Target Directives

### Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>after</strong></td>
</tr>
<tr>
<td><strong>Identification of</strong></td>
<td></td>
<td><strong>before</strong></td>
</tr>
<tr>
<td>constructs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>target</td>
<td>a task encounters a target</td>
<td>region is executed</td>
</tr>
<tr>
<td>data construct</td>
<td>region is executed</td>
<td></td>
</tr>
<tr>
<td>target data</td>
<td>a task encounters a target</td>
<td>new data environment is created</td>
</tr>
<tr>
<td>data construct</td>
<td>data construct</td>
<td></td>
</tr>
<tr>
<td>target update</td>
<td>a task encounters a target</td>
<td>data consistency is established</td>
</tr>
<tr>
<td>update construct</td>
<td>update construct</td>
<td></td>
</tr>
</tbody>
</table>
## OMPT: Begin Events for Target Directives

### Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification of constructs</td>
<td>target</td>
<td>after a task encounters a target region is executed</td>
</tr>
<tr>
<td></td>
<td>target data</td>
<td>before new data environment is created</td>
</tr>
<tr>
<td></td>
<td>target update</td>
<td>after a task encounters a target update construct</td>
</tr>
</tbody>
</table>

| Data transfer host <-> device | |
|------------------------------| |

**Tim Cramer** | **IT Center der RWTH Aachen University**
## Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OMPT: Begin Events for Target Directives</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Identification of constructs</strong></td>
<td>target</td>
<td>after</td>
</tr>
<tr>
<td></td>
<td>target data</td>
<td>before</td>
</tr>
<tr>
<td></td>
<td>target update</td>
<td>after</td>
</tr>
<tr>
<td><strong>Data transfer</strong></td>
<td>data map</td>
<td>-</td>
</tr>
<tr>
<td><strong>host &lt;-&gt; device</strong></td>
<td></td>
<td>before</td>
</tr>
</tbody>
</table>

- **target**: a task encounters a target construct
- **target data**: a task encounters a target data construct
- **target update**: a task encounters a target update construct
- **host <-> device**: a variable is mapped (event for each variable)
OMPT: Begin Events for Target Directives

### Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OMPT</strong></td>
<td><strong>Invocation</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td><strong>Event (begin)</strong></td>
<td><strong>Invocation</strong></td>
</tr>
<tr>
<td><strong>Identification of</strong></td>
<td><strong>Invocation</strong></td>
<td></td>
</tr>
<tr>
<td><strong>constructs</strong></td>
<td><strong>Event (begin)</strong></td>
<td><strong>Invocation</strong></td>
</tr>
<tr>
<td><strong>target</strong></td>
<td><strong>a task encounters a</strong></td>
<td><strong>region is executed</strong></td>
</tr>
<tr>
<td><strong>target data</strong></td>
<td><strong>a task encounters a target</strong></td>
<td><strong>new data environment is created</strong></td>
</tr>
<tr>
<td><strong>target update</strong></td>
<td><strong>a task encounters a target</strong></td>
<td><strong>data consistency is established</strong></td>
</tr>
<tr>
<td><strong>Data transfer</strong></td>
<td><strong>Data transfer</strong></td>
<td></td>
</tr>
<tr>
<td><strong>host &lt;-&gt; device</strong></td>
<td><strong>data map</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Kernel invocation</strong></td>
<td><strong>Kernel invocation</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identification of constructs</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>target</td>
<td>a task encounters a target region is executed</td>
<td></td>
</tr>
<tr>
<td>target data</td>
<td>a task encounters a target data construct</td>
<td>new data environment is created</td>
</tr>
<tr>
<td>target update</td>
<td>a task encounters a target update construct</td>
<td>data consistency is established</td>
</tr>
<tr>
<td><strong>Data transfer host &lt;-&gt; device</strong></td>
<td>data map</td>
<td>-</td>
</tr>
<tr>
<td><strong>Kernel invocation</strong></td>
<td>target invoke</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(in relation to an enclosing target region)</td>
<td></td>
</tr>
</tbody>
</table>

**OMPT: Begin Events for Target Directives**

- **Relevant information for performance analysis**
  - **OMPT**
  - **Begin Events**
  - **Target Directives**

- **Description**
  - **Event (begin)**
  - **Invocation**

- **Identification of constructs**
  - **target**
    - a task encounters a target region is executed
  - **target data**
    - a task encounters a target data construct
  - **target update**
    - a task encounters a target update construct

- **Data transfer host <-> device**
  - **data map**
    - -
  - **-**
    - a variable is mapped (event for each variable)

- **Kernel invocation**
  - **target invoke**
    - (in relation to an enclosing target region)

---

Performance Analysis for Target Devices with the OpenMP Tools Interface

Tim Cramer | IT Center der RWTH Aachen University
### Relevant information for performance analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Event (begin)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identification of constructs</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>target</td>
<td>a task encounters a target construct</td>
<td>region is executed</td>
</tr>
<tr>
<td>target data</td>
<td>a task encounters a target data construct</td>
<td>new data environment is created</td>
</tr>
<tr>
<td>target update</td>
<td>a task encounters a target update construct</td>
<td>data consistency is established</td>
</tr>
<tr>
<td><strong>Data transfer host &lt;-&gt; device</strong></td>
<td>data map</td>
<td>a variable is mapped (event for each variable)</td>
</tr>
<tr>
<td><strong>Kernel invocation</strong></td>
<td>target invoke (in relation to an enclosing target region)</td>
<td>target function is invoked on a device</td>
</tr>
</tbody>
</table>
OMPT: Signatures for Target Events

Signatures

- Only two different signature types needed for begin events

<table>
<thead>
<tr>
<th>Signature</th>
<th>Events</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>new target callback</td>
<td>target, target data, target update,</td>
<td>task id, target id, device id, target function</td>
</tr>
<tr>
<td></td>
<td>target invoke</td>
<td></td>
</tr>
<tr>
<td>new data map</td>
<td>data map</td>
<td>task id, target id, map id, device id, sync type, map type, data size</td>
</tr>
</tbody>
</table>

- Possible additional information for
  - target* events: source code line
    (In general: Missing source code information is a big disadvantage of OMPT compared to source-to-source instrumentation)
  - data map: variable name

- OpenMP 4.1: Each target* region will be an implicit task
  - target id will be obsolete
OMPT Inquiry Functions (1/2)

- **Prerequisite**
  
  → calling of OpenMP runtime library routine is unsafe (e.g., `omp_get_theread_num()`) → OMPT inquiry functions

  → all device inquiry functions are only allowed to be called in the extend of a

  → target data region

  → target update region

  → in the specified target event callbacks

  → undefined behavior otherwise
**OMPT Inquiry Functions (2/2)**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| OMPT_API ompt_target_device_id_t
  
  ompt_get_target_device_id(); | Returns the ID of the active target device. |
| OMPT_API ompt_target_id_t
  
  ompt_get_target_id(); | Returns the ID of the current target region. |
| OMPT_API ompt_target_device_time_t
  
  ompt_get_target_device_time (omp_target_device_id_t id); | Returns the current time stamp on the target device with ID id. (Can be used to synchronize time stamps on the target device with time stamps on the host.) |
| OMPT_API void ompt_target_map (void* dst, void* src, ompt_target_device_id_t id, ompt_data_map_t map_type, ompt_target_sync_t sync_type, ompt_data_size_t bytes); | Maps the specified number of bytes between host and target device according to the map type. (Can be used to map collected performance data / OMPT events from the device to the host.) |
OMPT: Prototype Implementation

- **Extension of the Intel-/LLVM-based OpenMP implementation**
  - OMPT 3.1 is part of the LLVM OpenMP runtime already [1]
  - OMPT 4.0 target support available as separated branch / fork [2]
  - Intel uses additional offload library (OpenMP library is not called for target* directives)
  - All target device events / signatures are implemented
  - Only the inquiry functions `ompt_get_device_time()` and `ompt_target_data_map()` are missing (work in progress)

- **Event Handling**
  - All target events: host-side only invocation
  - Analysis of “normal” OMPT events on a device
    - Callback registration on device possible (e.g., with preloading)
    - Tools developer are responsible for event handling (device and/or host)

[1] [http://openmp.llvm.org](http://openmp.llvm.org)
[2] [https://github.com/OpenMPToolsInterface](https://github.com/OpenMPToolsInterface)
OMPT: Tool Integration (1/3)

Using the OMPT extension in Score-P

![Diagram showing the integration of OMPT in Score-P and libmpti]

- **Score-P**: Tool infrastructure
- **libmpti**: OMPT instrumented
- **Control flow**
- **Data transfer**
OMPT: Tool Integration (2/3)

Visualization in Vampir

→ SPEC ACCEL Benchmark 314.omrig (ported to OpenMP 4.0)

- OpenMP target* regions on host
- OpenMP regions on the MIC
- Data mapping operations
- Region stack
- Context view

Performance Analysis for Target Devices with the OpenMP Tools Interface
Tim Cramer | IT Center der RWTH Aachen University
OMPT: Tool Integration (3/3)

OMPT + Score-P Overhead

→ Synthetic double buffering benchmark (maps 4 GB in 200 MB chunks)

<table>
<thead>
<tr>
<th></th>
<th>Runtime [s]</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uninstrumented</td>
<td>28.08</td>
<td>-</td>
</tr>
<tr>
<td>Score-P (host-sided events only)</td>
<td>28.50</td>
<td>1.5 %</td>
</tr>
<tr>
<td>Score-P (host-sided and devices sided events)</td>
<td>32.23</td>
<td>12.9 %</td>
</tr>
</tbody>
</table>

→ Overhead for host- and device-sided events is basically the data transfer
   → Only little influence on the application runtime itself

→ Used test system
   → Host: 2-socket Intel Sandybridge (Xeon E5-2650) CPUs
   → Target device: Intel Xeon Phi 5110P coprocessors
Summary & Conclusion

Extension of OMPT

- All OpenMP 4.x features should be covered in OMPT
- Proposal for target directives includes 10 new events, 4 signatures and 4 inquiry functions
- Prototype reference implementation in LLVM-based OpenMP runtime available
- Prototype support in Score-P measurement infrastructure available
- Source code information is missing in OMPT
- High detail level of information (size of mapped variables)
- Low overhead for proposed new events

Outlook

- Extend the OpenMP technical report to get proposed extension into the standard
Thank you for your attention.

Questions?