OpenMP Tools based on OMPT

Joachim Protze
OpenMP Booth, SC’19
Implementation state of OMPT

- LLVM: since 8.0
- icc: since 19.0.1

- OMPT support for target construct currently missing
OMPT support for sampling

- OMPT defines states like *barrier-wait, work-serial* or *work-parallel*
  - Allows to collect OMPT state statistics in the profile
  - Profile break down for different OMPT states

- OMPT provides frame information
  - Allows to identify OpenMP runtime frames.
  - Runtime frames can be eliminated from call trees

```c
void foo() {}
void baz() {#omp foo();}
int main()
{foo(); #omp foo();
baz(); return 0;}
```
HPCToolkit: OpenMP-aware callstack with OMPT
HPCToolkit’s Support for OpenMP TARGET

- HPCToolkit implementation of OMPT OpenMP API
  - host monitoring
    - leverages callbacks for regions, threads, tasks
    - employs OMPT API for call stack introspection
  - GPU monitoring
    - leverages callbacks for device initialization, kernel launch, data operations
    - reconstruction of user-level calling contexts
- Leverages implementation of OMPT in LLVM OpenMP and libomptarget

ECP QMCPACK Project: minigmc using OpenMP TARGET (Power9 + NVIDIA V100)

Reconstruct full calling contexts that include:
- Outlined procedures for OpenMP parallel regions
- Offloaded OpenMP TARGET computation and synchronization
Sampling vs. Instrumentation

**Sampling**
- Running program is periodically interrupted to take measurement
- *Statistical* inference of program behavior
- Works with unmodified executables

**Diagram**
- Sampling timeline with labeled functions and measurement points.

**Instrumentation**
- Every event of interest is captured directly
- More detailed and *exact* information
- Typically: recompile for instrumentation

**Diagram**
- Instrumentation timeline with detailed function execution and measurement points.
Scalable Performance Measurement Infrastructure for Parallel Codes
- Instrumenter, libraries, and tools to generate profile and trace measurements
- Provides measurement data for:

- Available under 3-clause BSD open-source license
- Documentation & sources: http://www.score-p.org
- Contact: support@score-p.org
Score-P and OMPT

- Implementation Challenge: Overdue events and non-deterministic scheduling, logical vs. system view, otherwise straightforward
- Similar performance compared to OPARI2
  - EPCC OpenMP micro-benchmark suite
  - SPEC OMP2012
- **OMPT** can serve as **drop-in replacement** for OPARI2 (work in progress)

IWOMP19 paper: Score-P and OMPT: Navigating the perils of callback-driven parallel runtime introspection [https://doi.org/10.1007/978-3-030-28596-8_2](https://doi.org/10.1007/978-3-030-28596-8_2)
TAU v2.29: OpenMP Support Update

- Built-in support for OMPT 5.0 specification
  
  ```
  $ ./configure -bfd=download -openmp -ompt ...
  ```

- Detects if compiler supports OMPT / OpenMP 5.0

- Compiles support for latest LLVM OpenMP runtime otherwise
  - Only for GCC, LLVM/CLANG, Intel compilers without support

- Developing target support

- Linker-based instrumentation (static executables)
  
  ```
  $ tau_cc.sh -o foo *.o libstuff.a
  ```

- Runtime preloading (dynamic executables)
  
  ```
  $ tau_exec -T openmp,ompt -ompt ./my_program
  ```
# TAU Profile Views

**pprof output:**

**Thread 0 callpath:**

<table>
<thead>
<tr>
<th>%Time</th>
<th>Exclusive msec</th>
<th>Inclusive msec</th>
<th>#Call</th>
<th>#Subs</th>
<th>Inclusive Name use/call</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0</td>
<td>8</td>
<td>1,038</td>
<td>1</td>
<td>1</td>
<td>.TAU application</td>
</tr>
<tr>
<td>99.6</td>
<td>0.963</td>
<td>1,029</td>
<td>1</td>
<td>1</td>
<td>main</td>
</tr>
<tr>
<td>95.6</td>
<td>0.247</td>
<td>1,029</td>
<td>1</td>
<td>1</td>
<td>do_work</td>
</tr>
<tr>
<td>95.4</td>
<td>2</td>
<td>1,048</td>
<td>5</td>
<td>5</td>
<td>OpenMP_Implicit_Task</td>
</tr>
<tr>
<td>52.7</td>
<td>0.108</td>
<td>1,021</td>
<td>1</td>
<td>1</td>
<td>compute</td>
</tr>
<tr>
<td>52.7</td>
<td>0.006</td>
<td>1,021</td>
<td>1</td>
<td>1</td>
<td>OpenMP_Parallel_Region</td>
</tr>
<tr>
<td>52.7</td>
<td>0.012</td>
<td>1,023</td>
<td>1</td>
<td>1</td>
<td>omp_outlined</td>
</tr>
<tr>
<td>52.7</td>
<td>0.006</td>
<td>1,023</td>
<td>1</td>
<td>1</td>
<td>omp_outlined_debug</td>
</tr>
<tr>
<td>42.5</td>
<td>0.039</td>
<td>823</td>
<td>1</td>
<td>1</td>
<td>compute_interchange</td>
</tr>
<tr>
<td>42.5</td>
<td>0.005</td>
<td>823</td>
<td>1</td>
<td>1</td>
<td>omp_outlined</td>
</tr>
<tr>
<td>42.5</td>
<td>0.037</td>
<td>822</td>
<td>1</td>
<td>1</td>
<td>allocateMatrix</td>
</tr>
<tr>
<td>42.5</td>
<td>0.022</td>
<td>822</td>
<td>1</td>
<td>1</td>
<td>freeMatrix</td>
</tr>
<tr>
<td>4.2</td>
<td>78</td>
<td>81</td>
<td>3</td>
<td>3</td>
<td>Initialize</td>
</tr>
<tr>
<td>0.2</td>
<td>0.399</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>omp_outlined</td>
</tr>
<tr>
<td>0.1</td>
<td>0.029</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>omp_outlined_debug</td>
</tr>
<tr>
<td>0.1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>allocateMatrix</td>
</tr>
<tr>
<td>0.0</td>
<td>0.331</td>
<td>0.331</td>
<td>3</td>
<td>3</td>
<td>freeMatrix</td>
</tr>
</tbody>
</table>

**Node 0; Context 0; Thread 0:**

**Node 0; Context 0; Thread 1:**

---

**Performance Measurement and Analysis using the TAU Performance System®**

U.S. DEPARTMENT OF ENERGY
Office of Science

11
Data race detection tool: Archer

- Error checking tool for
  - Memory errors
  - **Threading errors** (OpenMP, Pthreads)
- Based on ThreadSanitizer (runtime check)
- Available for Linux, Windows and Mac
- Supports C, C++
- Workflow available for Fortran
- Synchronization information based on **OMPT**

- More info: [https://github.com/PRUNERS/archer](https://github.com/PRUNERS/archer)

- Will be part of the next release of LLVM: 9.1 / 10.0
  - Accepted and committed to LLVM master branch
More OpenMP Correctness Tools

• ROMP: Yizi Gu, John M. Mellor-Crummey: Dynamic data race detection for OpenMP programs. SC 2018

• TaskSanitizer: Hassan Salehe Matar, Didem Unat: Runtime Determinacy Race Detection for OpenMP Tasks. Euro-Par 2018

• Ricardo Bispo Vieira, Antoine Capra, Patrick Carribault, Julien Jaeger, Marc Pérache, Adrien Roussel: Detecting Non-sibling Dependencies in OpenMP Task-Based Applications. IWOMP 2019
Tools for OpenMP Tasking

- Vishakha Agrawal, Michael J. Voss, Pablo Reble, Vasanth Tovinkere, Jeff Hammond and Michael Klemm
  Visualization of OpenMP Task Dependencies using Intel Advisor Flow Graph Analyzer, IWOMP 2018

- Peder Voldnes Langdal, Magnus Jahre and Ananya Muddukrishna:
  Extending OMPT to support Grain Graphs, IWOMP 2017