Evaluating Support for OpenMP Offload Features

Jose Monsalve Diaz
Swaroop Pophale
Kyle Friedline

Oscar Hernandez
David E. Bernholdt
Sunita Chandrasekaran
Outline

1. Introduction
2. Problem statement and contributions
3. OpenMP 4.5 offloading
4. Methodology
5. Experimental Setup
6. Results
7. Conclusions
Introduction
Current Trends in HPC
(Data from Top 500 list: www.top500.org)
Current Trends in HPC
(Data from Top 500 list: www.top500.org)
Why Accelerator devices?

- Power wall
- Simpler cores and considerably larger core count
- Programs taking advantage of SIMD and SPMD models
  - Better performance/Watt ratio
  - Benefit from large parallelism

But there are still programmability challenges...
Problem statement and contributions
Problem statement

Given the specifications of OpenMP 4.5, the multiple compiler implementations that exist, and the different systems where this programmer model will be used. how to assess the level of compliance of implementations and systems with respect to the specifications document?
Problem statement

Explanation

Programming model specifications document

“Legal” document that binds the implementation and the user

- Compiler developers guide their products based on the specifications documents. They must respect them to claim support
- Users do not need to learn implementation specific aspects of the programming model. They use the specifications
- What about the system that the user is running on?
Programming Model Specifications

Problem Model User

Compiler implementer

System where User is running

Uses specs as guide for implementation

Uses vendor’s compiler to support programming model

Uses system as programming platform

Uses specs as guide for implementation

Uses specs as guide for programming
Problem statement

Explanation

Programming Model Specifications

Uses specs as guide for implementation

Compiler implementer

Uses vendor’s compiler to support programming model

System where User is running

Uses system as programming platform

Programming Model User

Uses specs as guide for programming

How to guarantee an implementation compliance with the specs?

How to guarantee a system compliance with the compiler and programming model?

P2S2 2018 - University of Oregon at Eugene, OR - Presented by: Jose Monsalve (josem@udel.edu)
Contributions of this work

We are the “lawyers” of the OpenMP 4.5 Specifications:

- Identify extent of OpenMP offload support (target directives) in OpenMP 4.5 implementations such as GCC, Clang/LLVM, IBM XL and Cray CCE
- Analyze support for common code kernels identified across a range of DOE applications and test their support across all accessible OpenMP 4.5 implementations
- Identify and report inconsistencies or bugs in specific implementations to their respective compiler developers
- Present performance data for different directives across different OpenMP 4.5 implementations
OpenMP 4.5 Offloading
OpenMP 4.5 Machine model

Host Memory

INTERCONNECT

Device memory

Target

Device 1

Device memory

Target

Device n
OpenMP 4.5 Offloading
Code Execution Model

Host

Memory

Code

Device memory

Target

Device 1

\ldots \ldots \ldots \ldots

Device memory

Target

Device n
OpenMP 4.5 Offloading
Code Execution Model

Host

Memory

Device memory

Device 1

Target

Device n

Target

Code

Host

Memory
OpenMP 4.5 Offloading
Code Execution Model
OpenMP 4.5 Offloading
Data management model
OpenMP 4.5 Offloading
Data management model
OpenMP 4.5 Offloading
Data management model

Host

Memory

Device memory

Data

Target

Device 1

... ...

Device n

Memory
OpenMP 4.5 Offloading

Data management model

Host

Memory

Target

Device 1

Device n

Device memory

Device memory

Data
OpenMP 4.5 Offloading
Data management model
OpenMP Offloading programming model

```
#omp target
// target region

#omp teams
// distributed among teams

#omp distribute
for (i:0:N)

#omp parallel for
for (j:0:M)

#omp simd
for (k:0:L) { ... }
```
Methodology
Validation of OpenMP 4.5 offloading features

Workflow

- Analyze OpenMP 4.5 offload directive OR ECP Application
- Formulate test
- Discuss validity and adherence to specification
- Test valid?
  - YES
  - NO
- Test accepted?
  - NO
  - YES
- Test passes?
  - YES
  - Open for community review
  - File Bug report with vendor
  - Add to the V&V suite
  - NO
- Why?
  - Implementation Bug
  - Specification issue
- Bring to OpenMP Specification discussion
Validation of OpenMP 4.5 offloading features

Example C Code

```
int num_dev = omp_get_num_devices();
int h_matrix [N*num_dev];

for (int dev = 0; dev < num_dev; ++dev) {
    #pragma omp target data map(from: h_matrix[dev*N:N]) device(dev)
    {
        #pragma omp target map(from: h_matrix[dev*N:N]) device(dev)
        {
            for (int i = 0; i < N; ++i)
                h_matrix[dev*N + i] = dev;
        }
    }
}

// checking results
for (int dev = 0; dev < num_dev; ++dev) {
    for (int i = dev*N ; i < (dev+1)*N; ++i)
        OMPVV_TEST(errors, dev != h_matrix[i]);
}
```

Get number of available devices
Map data from selected device to host
Create target region on selected device
Computation offloaded to selected device
Checking data mapping and code offloading on multiple devices
Compare results with expected
Comparing runtime overhead

pseudocode

```c
OMPVV_INIT_TEST;
for ( i = 0 ; i < NUM_REP ; i ++ ) {
    OMPVV_START_TIMER;
    #pragma omp ...
    OMPVV_TEST_LOAD; // if necessary
    OMPVV_STOP_TIMER;
    OMPVV_REGISTER_TEST;
}
OMPVV_PRINT_RESULT;
```
# Experimental setup

<table>
<thead>
<tr>
<th>System</th>
<th>Model</th>
<th>Processors</th>
<th>Cores/node</th>
<th>Threads/node</th>
<th>Memory</th>
<th>Accelerator</th>
<th>Compilers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titan</td>
<td>Cray XK7</td>
<td>AMD Opteron</td>
<td>16</td>
<td>16</td>
<td>32 GB</td>
<td>1 NVIDIA K20X</td>
<td>CCE 8.7.2</td>
</tr>
<tr>
<td>Summitdev</td>
<td>IBM S822LC</td>
<td>2x Power8</td>
<td>20</td>
<td>160</td>
<td>256 GB</td>
<td>4 NVIDIA P100</td>
<td>GCC 7.1.1, Clang 3.8.0, XLC 13.1.6</td>
</tr>
<tr>
<td>Summit</td>
<td>IBM AC922</td>
<td>2x Power9</td>
<td>42</td>
<td>168</td>
<td>512 GB</td>
<td>6 NVIDIA V100</td>
<td>Clang 3.8.0, XLC 13.1.7</td>
</tr>
<tr>
<td>In-House</td>
<td>Generic</td>
<td>2x Intel Xeon E5-2670</td>
<td>16</td>
<td>32</td>
<td>64 GB</td>
<td>1 NVIDIA K20X</td>
<td>Clang 7.0.0, GCC 8.1.0</td>
</tr>
</tbody>
</table>
Experimental setup
Summit’s Node
Results

* new results since paper publication
## Summary of compiler supported features

<table>
<thead>
<tr>
<th>OpenMP Feature</th>
<th>GCC 7.1.1</th>
<th>Summitdev gfortran 7.1.1*</th>
<th>Clang 3.8.0</th>
<th>XLC 13.1.6</th>
<th>XLF* 15.1.7</th>
<th>Summit Clang 3.8.0</th>
<th>XLC 13.1.7</th>
<th>Titan CCE 8.7.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>14/14</td>
<td>13/13</td>
<td>14/14</td>
<td>13/14</td>
<td>12/13</td>
<td>12/14</td>
<td>11/14</td>
<td>13/14</td>
</tr>
<tr>
<td>target data</td>
<td>5/6</td>
<td>4/4</td>
<td>6/6</td>
<td>6/6</td>
<td>2/4</td>
<td>6/6</td>
<td>6/6</td>
<td>3/6</td>
</tr>
<tr>
<td>target enter/exit data</td>
<td>6/7</td>
<td>-</td>
<td>6/7</td>
<td>6/7</td>
<td>-</td>
<td>6/7</td>
<td>6/7</td>
<td>5/7</td>
</tr>
<tr>
<td>target enter data</td>
<td>6/7</td>
<td>-</td>
<td>6/7</td>
<td>6/7</td>
<td>-</td>
<td>6/7</td>
<td>6/7</td>
<td>5/7</td>
</tr>
<tr>
<td>target update</td>
<td>5/5</td>
<td>-</td>
<td>5/5</td>
<td>4/5</td>
<td>-</td>
<td>5/5</td>
<td>4/5</td>
<td>4/5</td>
</tr>
<tr>
<td>target teams distribute</td>
<td>10/11</td>
<td>-</td>
<td>8/11</td>
<td>10/11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>9/11</td>
</tr>
<tr>
<td>target teams distribute parallel for</td>
<td>13/14</td>
<td>-</td>
<td>11/14</td>
<td>11/14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10/14</td>
</tr>
</tbody>
</table>
Example of problematic features
XL and defaultmap

Specifications (Section 2.15.5 - page 216):

4 ● If a `defaultmap(tofrom: scalar)` clause is not present then a scalar variable is not mapped, but instead has an implicit data-sharing attribute of firstprivate (see Section 2.15.1.1 on page 179).

Test:

```c
enum { VAL1 = 1, VAL2, VAL3, VAL4 } scalar_enum = VAL1
#pragma omp target
{
    scalar_enum = VAL4;
}
OMPVV_TEST_AND_SET_VERBOSE(errors, scalar_enum != VAL1);
```

Failed condition
# Target directive

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>309.99</td>
</tr>
<tr>
<td>target defaultmap</td>
<td>308.90</td>
</tr>
<tr>
<td>target map from</td>
<td>303.22</td>
</tr>
<tr>
<td>target map to</td>
<td>304.99</td>
</tr>
<tr>
<td>target map tofrom</td>
<td>302.62</td>
</tr>
</tbody>
</table>

- **GCC (8.1.0 on x86 system)**
- **CLANG Trunk 7.0.0 (x86 system)**
- **XLC (13.1.6 on Summitdev)**
- **CLANG (3.8.0 on Summitdev)**
- **GCC (7.1.1 on summitdev)**
Target directive

- target
- target depend
- target device
- target firstprivate
- target if
- target is device ptr
- target private

Time (us)

0  100  200  300  400  500  600  700  800  900  1000  1100
Target data directive

- **target data map from**:
  - GCC(8.1.0 on x86 system)
  - CLANG Trunk 7.0.0 (x86 system)

- **target data map to**:
  - XLC (13.1.6 on Summitdev)
  - CLANG (3.8.0 on Summitdev)
  - GCC (7.1.1 on summitdev)

- **target data device**:
  - 22.18
  - 23.11

- **target data if**:
  - 22.19
  - 23.05

- **target data map tofrom**:
  - 22.21
  - 23.25

**Time (us)**: 0 100 200 300 400 500 600 700 800 900 1000 1100
Target enter/exit data directive

- Target enter data map if false
- Target enter data map alloc
- Target enter data map depend
- Target enter data map device
- Target enter data map if true
- Target enter data map to

Time (us)

GCC (8.1.0 on x86 system)
CLANG Trunk 7.0.0 (x86 system)
XLC (13.1.6 on Summitdev)
CLANG (3.8.0 on Summitdev)
GCC (7.1.1 on summitdev)
Target enter/exit data directive

- Target exit data map if false
- Target exit data map delete
- Target exit data map depend
- Target exit data map device
- Target exit data map from
- Target exit data map if true

Time (us)

GCC (8.1.0 on x86 system)
CLANG Trunk 7.0.0 (x86 system)
XLC (13.1.6 on Summitdev)
CLANG (3.8.0 on Summitdev)
GCC (7.1.1 on summitdev)
Conclusions
Conclusions

- We have shown how use of accelerators and heterogeneous systems are the current trend in HPC and most likely will continue to increase.
- As a result programming models are fastly adapting to these new architectural features.
- We have shown the importance of a proper methodology to assess support and status of current compiler implementations (GCC, Clang/LLVM, IBM XL and Cray CCE) and the latest DOE Systems.
- We have shown a possible methodology, and the results of applying it to the OpenMP 4.5 specifications, emphasizing offloading features identified in DOE applications.
- As the development continues, we have seen compiler developers fastly adapting and responding to bug reports. We appreciate their effort and responsiveness.
Visit our website
https://crpl.cis.udel.edu/ompvvsollve/

Work supported by the U.S. Department of Energy, Office of Science, the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration under contract number DE-AC05-00OR22725.

Contact information:
Jose Monsalve (josem@udel.edu)
Swaroop Pophale (pophaiess@ornl.gov)
Kyle Friedline (utimatu@udel.edu)
Oscar Hernandez (oscar@ornl.gov)
Sunita Chandrasekaran (schandra@udel.edu)
Copyright


Logos under CC0:

Back up slides
Top 500: Average number of cores per socket
Top 500: Number of systems with accelerators

0 0 1 1 1 4 8 5 7 9 17 19 39 58 62 54 53 64 73 88 103 94 86 91 102 110

NVIDIA  Other  Intel  AMD  total
Top 500: Number of systems with accelerators
Validation of OpenMP 4.5 offloading features

Design and write tests for OpenMP 4.5 Offloading features:

1. Study the specifications
2. Formulate testing methodology
3. Write initial test implementation
4. Discuss tests with team
5. Make corrections, report bugs or request clarifications
6. Run and report test result on multiple compilers available to us
   a. Main focus on DOE Systems