OpenACC 2.5 Validation Testsuite targeting multiple architectures

Kyle Friedline, Sunita Chandrasekaran, University of Delaware, USA
Graham Lopez, Oscar Hernandez, Oak Ridge National Lab, USA
June 22, 2017
P3MA Workshop @ ISC

https://github.com/OpenACCUserGroup/OpenACCV-V
OpenACC- a directive-based parallel programming model

Add Simple Compiler Directive

```c
main()
{
    <serial code>
    #pragma acc kernels
    {
        <parallel code>
    }
}
```
Motivation

• To help developers find and fix compiler bugs
• To assist users to know where the compilers stand in terms of OpenACC functionality support
• To assist users with fundamental understanding of spec features
• To discern ambiguities in the OpenACC specifications
• To discuss misinterpretations of feature definitions
Top HPC Apps adopting OpenACC

- ANSYS Fluent
- Gaussian
- VASP

3 of Top 10 Apps

- GTC
- XGC
- ACME
- FLASH
- LSDalton

5 ORNL CAAR Codes

- COSMO
- ELEPHANT
- RAMSES
- ICON
- ORB5

5 CSCS Codes
OpenACC

• Execution:
  • The `parallel` directive for explicit parallelization mapping
  • The `kernels` directive for implicit parallelization mapping
  • The `loop` construct for parallelization specification

• Memory:
  • The `data` directive for scoped, explicit-lifetime data elements
  • The `enter/exit` data directives for executable data transfer
OpenACC Execution Model

• Parallel
  • Best when given explicit parallelization dimensions.

• Kernels
  • Best when compiler is allowed to implement its determined best parallelization dimensions
  • Potential for multiple compute kernels allowing multiple, varying-size compute-resource allocations.

• Loop
  • Describes parallelization across for-loops with the addition of gang, worker, or vector clauses.
  • Describes data dependencies both in code structure and in loop iterations through the async, independent, and seq clauses.
OpenACC Memory Model

• Enter/Exit data directives
  • Executes memory allocation, deallocation, and transfer
  • Allows for cross-scope data lifetimes
  • Requires more vigilance on the part of the programmer to manage data lifetimes, especially with dynamic execution patterns

• Data region
  • Gives data a specified beginning and ending on device
  • Allows compiler to predetermine data presence via scopes and call trees.
  • Can lead to data remaining allocated on device longer than necessary to complete scope

• Data management clauses
  • Private
  • Firstprivate
  • Reduction
  • Cache
  • Deviceptr
  • Present
Multi-Paradigm Accelerating Model

• Multiple Accelerator Support
  • acc_get_num_devices
  • acc_set_device_type
  • acc_set_device_num
Compilers supporting OpenACC

• PGI
  • NVIDIA GPU, X86 CPU, X86 Xeon Phi, POWER
• GNU
  • NVIDIA GPU
• Cray
  • NVIDIA GPU
• Sunway OpenACC*
• Omni (PEZY-SC) from University of Tsukaba
• OpenARC from ORNL, OpenUH from UH and SBU

https://www.openacc.org/tools
The Validation Test Suite

• The suite of 177 tests includes 86 Fortran tests and 91 C tests covering:
  • All of the parallel construct and its clauses, but missing some fringe functionality and clause combinations
  • Most of kernels construct and its clauses, but missing all fringe functionality and clause combinations
  • Data directives (data, enter data, exit data) with primary clauses tested, but missing all fringe functionality and most clause combinations

• Tests build upon each other; initial tests test core functionality, while later tests built upon previously tested functionality
Test Suite Structure

• Primary test types
  • Unit tests
  • Incremental Integration Tests
  • Use Cases

• Test Design Issues
  • Backwards-compatibility
  • Compute complexity
  • Platform-independent test compatibility
Test Design

- Design issues addressed
  - Data clauses are called as 2.5 specifies
  - Compute complexity is kept minimal
  - Internal testing determines memory type for some platform dependent testing
Contributions of the Test Suite

• Check for correctness of implementations of OpenACC features
• Reported bugs to both PGI and GCC
• Check and report ambiguities or misinterpretation in the OpenACC specification
Open-sourcing V&V Suite

• Validation and Verification Suite will soon be made available to everyone
  • Target deadline late August 2017
• [https://github.com/OpenACCUserGroup/OpenACCV-V](https://github.com/OpenACCUserGroup/OpenACCV-V)
• Feedback on the suite is welcome
• Contributions to the suite welcome
• BSD 3-Clause license
# Tested Platforms

<table>
<thead>
<tr>
<th>K20</th>
<th>K80</th>
<th>P100</th>
<th>Ivy Bridge</th>
<th>Bulldozer</th>
<th>Power8+</th>
<th>Knights Landing</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGI</td>
<td>V16.10</td>
<td>V16.10*</td>
<td>V17.1</td>
<td>V16.10**/17.3</td>
<td>V16.10</td>
<td>V17.1</td>
</tr>
<tr>
<td>GNU</td>
<td>V6.3-20 170303</td>
<td>V6.0.0-20 160415</td>
<td>V6.0.0-20 160415</td>
<td>V6.3-201 70303</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*K80 was run against many versions of PGI which will be seen in a following slide.

**Version 16.10 reported in preference of 17.3 since 16.10 was released as the community edition. Freely available at no cost for download.

***Knights Landing is not officially supported; testing was performed by using –ta=haswell flag.
Results from PGI and GNU compilers

<table>
<thead>
<tr>
<th>Architecture</th>
<th>PGI Pass Rate</th>
<th>GNU Pass Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>K20</td>
<td>175/177</td>
<td>112/177</td>
</tr>
<tr>
<td>K80</td>
<td>175/177</td>
<td>113/177</td>
</tr>
<tr>
<td>Ivy Bridge</td>
<td>171/177</td>
<td>154/177</td>
</tr>
<tr>
<td>Bulldozer</td>
<td>172/177</td>
<td>157/177</td>
</tr>
</tbody>
</table>
# Cross-Platform Performance of PGI

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Pass Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>K20</td>
<td>175/177</td>
</tr>
<tr>
<td>K80</td>
<td>175/177</td>
</tr>
<tr>
<td>P100</td>
<td>175/177</td>
</tr>
<tr>
<td>Ivy Bridge</td>
<td>171/177</td>
</tr>
<tr>
<td>Bulldozer</td>
<td>172/177</td>
</tr>
<tr>
<td>Power8+</td>
<td>165/177</td>
</tr>
<tr>
<td>Knights Landing</td>
<td>167/177</td>
</tr>
</tbody>
</table>
## PGI Improvement Over Versions

<table>
<thead>
<tr>
<th>Compiler Version</th>
<th>Fortran Pass Rate</th>
<th>C Pass Rate</th>
<th>Fortran % Passed</th>
<th>C % Passed</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.10</td>
<td>60/86</td>
<td>67/91</td>
<td>69.8</td>
<td>73.6</td>
</tr>
<tr>
<td>15.1</td>
<td>64/86</td>
<td>80/91</td>
<td>74.4</td>
<td>87.9</td>
</tr>
<tr>
<td>15.5</td>
<td>65/86</td>
<td>80/91</td>
<td>95.6</td>
<td>87.9</td>
</tr>
<tr>
<td>15.10</td>
<td>68/86</td>
<td>84/91</td>
<td>79.1</td>
<td>92.3</td>
</tr>
<tr>
<td>16.1</td>
<td>69/86</td>
<td>84/91</td>
<td>80.2</td>
<td>92.3</td>
</tr>
<tr>
<td>16.4</td>
<td>82/86</td>
<td>84/91</td>
<td>95.3</td>
<td>92.3</td>
</tr>
<tr>
<td>16.7</td>
<td>85/86</td>
<td>90/91</td>
<td>98.8</td>
<td>98.9</td>
</tr>
<tr>
<td>16.10</td>
<td>85/86</td>
<td>90/91</td>
<td>98.8</td>
<td>98.9</td>
</tr>
<tr>
<td>17.1</td>
<td>85/86</td>
<td>90/91</td>
<td>98.8</td>
<td>98.9</td>
</tr>
<tr>
<td>17.3</td>
<td>85/86</td>
<td>90/91</td>
<td>98.8</td>
<td>98.9</td>
</tr>
</tbody>
</table>

These results are from running against K80
Future Work

• Near future work – open source V&V suite
• Improve documentation
  • Forward and backward referencing
• Addition of use-cases
• Build a more comprehensive suite (cross, orphan test cases)
• Keep up with the growing feature set of the specification
• Brainstorming building an infrastructure common to both OpenMP and OpenACC V&V