Overview of Model & Testsuite

OpenACC
- Specification is constantly evolving to accommodate the needs of application developers.

OpenACC Validation and Verification test suites is a Third-Party validation tool that evaluates compilers’ compliance with the specification and identifies ambiguities in the specification.

Infrastructure

- Made to run the tests with customization and simplicity, it can be tailored to any system through the configuration file.
- The most important features to the average user of the infrastructure are compiler and compiler flag selection, output format specification, conditional compilation, and hang-time limits.
- A comprehensive list of all of the options can be found here: https://github.com/OpenACCUserGroup/OpenACCV-V

Challenges

1. Code compilation and targeting device differ w/ Compilers - Systems
2. Code compilation and targeting device differ w/ Compilers - Systems
3. Code compilation and targeting device differ w/ Compilers - Systems
4. Few tests based on features yet to be implemented in compilers
5. Certain features can only be tested by stressing to failure
6. Some features are difficult to test based on the definition
7. Code compilation and targeting device differ w/ Compilers - Systems

Contribution to Community

- This Work Contributes:
  - Example of a challenging test: Init Directive w/ If Clause
  - Few tests based on features yet to be implemented in compilers
  - Certain features can only be tested by stressing to failure
  - Some features are difficult to test based on the definition

Results

![Fortran Tests](image1)

![C Tests](image2)

Analysis of Results

- The purpose is to validate and verify the implementations outlined by the OpenACC specification for all systems. As seen within the result section, many of the features have failed the tests: serial loop, atomic capture, routine bind functions with lambda functions, and the combination of the data clause copyin and copyout, to list a few.
- The C++ tests are not fully supported within the infrastructure, and there are plans to comb through to resolve this issue.
- Future goals are to continue developing tests for new features and to cover more edge cases. Using examples, we will also create an example guide that will give a beginner-friendly breakdown of how to implement OpenACC.

Infrastructure

- Provides a Testing Infrastructure that: evaluates compilers’ compliance with the OpenACC Validation and Verification testsuites
- Provides a Testing Infrastructure that: evaluates compilers’ compliance with the OpenACC Validation and Verification testsuites
- Provides a Testing Infrastructure that: evaluates compilers’ compliance with the OpenACC Validation and Verification testsuites
- Provides a Testing Infrastructure that: evaluates compilers’ compliance with the OpenACC Validation and Verification testsuites

Conclusion - Future Work

- Our future goals are to continue developing tests for new features and to cover more edge cases. Using examples, we will also create an example guide that will give a beginner-friendly breakdown of how to implement OpenACC.

Acknowledgments/ References

This material is based upon work supported by the National Science Foundation (NSF) under grant no. 1804699. We would also like to thank immensely OpenACC for their tremendous support towards GPUs, technical support and funding.

1. OpenACC Specifications. [https://www.openacc.org/specification]

Analysis of Results

- Compilers have different features implemented
- Improvements occur with newer versions of compilers
- Results are primarily consistent across architectures
- Provides a metric to measure future versions against
- Some compilers translate OpenACC to OpenMP, where some features are not equivalent between both models

LVM Integration

- Using the OpenACC V&V testsuite would be an objective, third-party assessment of LLVM’s (CLACC) conformance to the OpenACC specification that can be compared with other OpenACC implementations
- Certain compiler translators translate OpenACC to OpenMP to build upon the OpenMP support being developed for Clang and LLVM. A CMakeList file has been included in this project which is the entry point to our llvm-test-suite. It generates the Makefile which allows users to compile, run, and report test results. LLVM OpenACC V&V Integration found here: [https://github.com/llvm-doe-org/llvm-test-suite/tree/llvm.org/main/External/openacc_vv]

LLVM's (CLACC) Integration found here: [https://github.com/llvm-doe-org/llvm-test-suite/tree/llvm.org/main/External/openacc_vv]

Example of a challenging test: Init Directive w/ If Clause

- Few tests based on features yet to be implemented in compilers
- Certain features can only be tested by stressing to failure
- Some features are difficult to test based on the definition

Conclusion - Future Work

- The purpose is to validate and verify the implementations outlined by the OpenACC specification for all systems. As seen within the result section, many of the features have failed the tests: serial loop, atomic capture, routine bind functions with lambda functions, and the combination of the data clause copyin and copyout, to list a few.
- The C++ tests are not fully supported within the infrastructure, and there are plans to comb through to resolve this issue.
- Future goals are to continue developing tests for new features and to cover more edge cases. Using examples, we will also create an example guide that will give a beginner-friendly breakdown of how to implement OpenACC.