SOLLVE OpenMP V&V: The Team

ECP Project WBS 2.3.1.13 STPM15

Swaroop Pophale
David E. Bernholdt

Josh Davis
Thomas Huber
Sunita Chandrasekaran
Outline

• Introduction
  • What is SOLLVE?
  • Scope and Intent of the Validation and Verification Suite
  • OpenMP Offload (4.x+) Machine Model
  • OpenMP 5.0 New Features

• Validation and Verification Suite
  • Test design process
  • Examples
  • Infrastructure design

• Sample Results

• Success Stories

• Ways to collaborate
Introduction: The SOLLVE ECP Project
Introduction: Scope and Intent of V&V suite

• Verify the status of OpenMP implementations across ECP platforms
• Evaluation of OpenMP functionality and performance on target architectures
• Check implementations’ conformance to the OpenMP standard
• Tests and kernels primarily motivated by ECP Applications
• Verify platforms are ready for OpenMP applications
Introduction: OpenMP Offload (4.x+)
Machine Model

HOST

DEVICES

Device 1
Memory

Device N
Memory

Interconnect
Introduction: OpenMP 5.0 New Features

• OpenMP contexts, metadirective, and declare variant
• Addition of requires directive, including support for unified shared memory
• Memory allocators and support for deep memory hierarchies
• Descriptive loop construct
• Release/acquire semantics added to memory model
• First (OMPT) and third (OMPD) party tool support
• Completed support for Fortran 2003
• Added support for Fortran 2008, C11, C++11, C++14 and C++17
V&V: Test Design Process

1. Analyze OpenMP directive OR ECP Application
2. Formulate test
3. Discuss validity and adherence to specification
   - Valid test? NO
   - YES
4. Test with available implementations
5. Test passes? NO
6. Open for review
7. Test accepted? NO
8. Why?
   - Specification issue
   - Implementation Bug
9. Bring to OpenMP Specification discussion
10. File Bug report with vendor
11. Add to the V&V suite
12. YES
V&V Example 1: Understanding Spec.

```c
#pragma omp target data map(tofrom: a[0:ARRAY_SIZE]) {
  #pragma omp target teams distribute
  ...
}

#pragma omp target data map(tofrom: a[0:ARRAY_SIZE/2]) {
  #pragma omp target teams distribute
  ...
}
```
V&V Example 1: Understanding Spec.

```c
#pragma omp target data map(tofrom: a[0:ARRAY_SIZE]) {
    #pragma omp target teams distribute
    ...
}
```

```c
#pragma omp target data map(tofrom: a[0:ARRAY_SIZE/2]) {
    #pragma omp target teams distribute
    ...
}
```
V&V Example 1: Understanding Spec.

```c
#pragma omp target data map(tofrom: a[0:ARRAY_SIZE]) {
#pragma omp target teams distribute map(alloc: a[0:ARRAY_SIZE])
    ...
}
```

```c
#pragma omp target data map(tofrom: a[0:ARRAY_SIZE/2]) {
#pragma omp target teams distribute map(alloc: a[0:ARRAY_SIZE/2])
    ...
}
```

- Specification doesn’t explicitly mention this requirement, but it is an implication arising from multiple interacting sections of the specification.
- Prone to misinterpretation by vendors — XLC compiles and runs with no complaint, Clang gives a compile error explaining the problem, and GCC gives a runtime error.
V&V Example 2: Ambiguous Specification

Restrictions

- A list item cannot appear in both a `map` clause and a data-sharing attribute clause on the same construct.

(From map clause specification, pg. 218)

```
#pragma omp target teams distribute shared(shared) map(tofrom:shared)
for (int x = 0; x < SIZE; ++x) {
    ...
}

#pragma omp target teams distribute shared(shared) defaultmap(tofrom:scalar)
for (int x = 0; x < SIZE; ++x) {
    ...
}
```
V&V Example 3: Test from Application

```c
#pragma omp target map(tofrom: a, sum) depend(out: a) nowait
{
    for (i = 0; i < N; i++) {
        sum++;
    }
    a += 1;
}

#pragma omp task depend(in: a) shared(a,errors)
{
    if(a != 1) {
        errors += 1;
    }
}
#pragma omp taskwait
if (sum != N) {
    errors++;  
}
```
V&V: Infrastructure Design

• Our infrastructure is based on a **Makefile + scripts**

• Design parameters:
  • Portability across multiple **compilers** and **systems** and easy to use
    • Support for different compiler options
    • Support for Lua-like Modules and batch schedulers
  • Fast test addition and modification
  • Divided compilation and execution phases
  • Subset of tests selection for partial execution
V&V: Infrastructure Design (cont.)

Test suite

- Tests
- ompvv
- sys
- template
- Makefile

- Target
data

- Target
team

distribute

- Results_template
- Make
- Scripts
- Systems

- Summit.def
- Cori.def
- Your_system.def

4.5
5.0
Sample Results: Summit
Sample Results: Cori
Sample Results: Current status of OMP Compilers

https://crpl.cis.udel.edu/ompvvsollve/

OPENMP VALIDATION AND VERIFICATION

This website contains all, related to the OpenMP Validation and Verification suite developed as part of the Exascale Computing Project (ECP). In particular, the Scaling OpenMP Via LUMP for Exascale Performance and Portability (SOLVE) project.

This project is a collaboration of:

- Oak Ridge National Laboratory
- University of Delaware
- ECP (Exascale Computing Project)
Success Stories

• Cray and AMD are actively using SOLLVE V&V tests to verify correctness and coverage of new features introduced in OpenMP 4.5 and 5.0 (on-going effort).

• Vendors are actively engaging to improve coverage and bug fixes.

• Implementation bugs were identified in the GCC, IBM, and Cray implementations and have been brought to the vendor’s attention.

• OLCF is using SOLLVE V&V as part of acceptance tests for exascale ECP platform Frontier.

• NERSC has integrated SOLLVE V&V for testing OpenMP 4.5/5.0 implementations.
Ways to Collaborate: GitHub

https://github.com/SOLLVE/sollve_vv
Ways to Collaborate: GitHub

• Need to check the behavior of YOUR OpenMP application?
  • Create an application kernel
  • SUBMIT an Issue OR CREATE a PR on https://github.com/SOLLVE/sollve_vv

• Found something we missed?
  • SUBMIT an Issue OR CREATE a PR on https://github.com/SOLLVE/sollve_vv
Ways to Collaborate: Contact information

Swaroop Pophale (pophale@ornl.gov)
David E. Bernholdt (bernholdtde@ornl.gov)
Josh Davis (jhdavis@udel.edu)
Thomas Huber (thuber@udel.edu)
Sunita Chandrasekaran (schandra@udel.edu)
Acknowledgement

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**.
For the OpenMP specification, tutorials, forum, reference guides, and links to other resources, visit www.openmp.org