

OpenMP Offloading Verification and Validation: Workflow and Road to 5.0

Thomas Huber & Joshua Davis (UD)

Jose Monsalve Diaz (UD)

Swaroop Pophale (ORNL)

Sunita Chandrasekaran (UD)¹

Kyle Friedline (UD)

Oscar Hernandez (ORNL)

David E. Bernholdt (ORNL)

¹schandra@udel.edu

https://github.com/SOLLVE/sollve_vv — Presented 14 April 2020

Outline

- Problem and motivation
- Current status of the test suite
- Next steps for 5.0 and new directions
 - Performance measurement
 - Continuous integration
 - Hardware detection
- Quick guide of running the test suite on a system and reporting results
- Conclusions

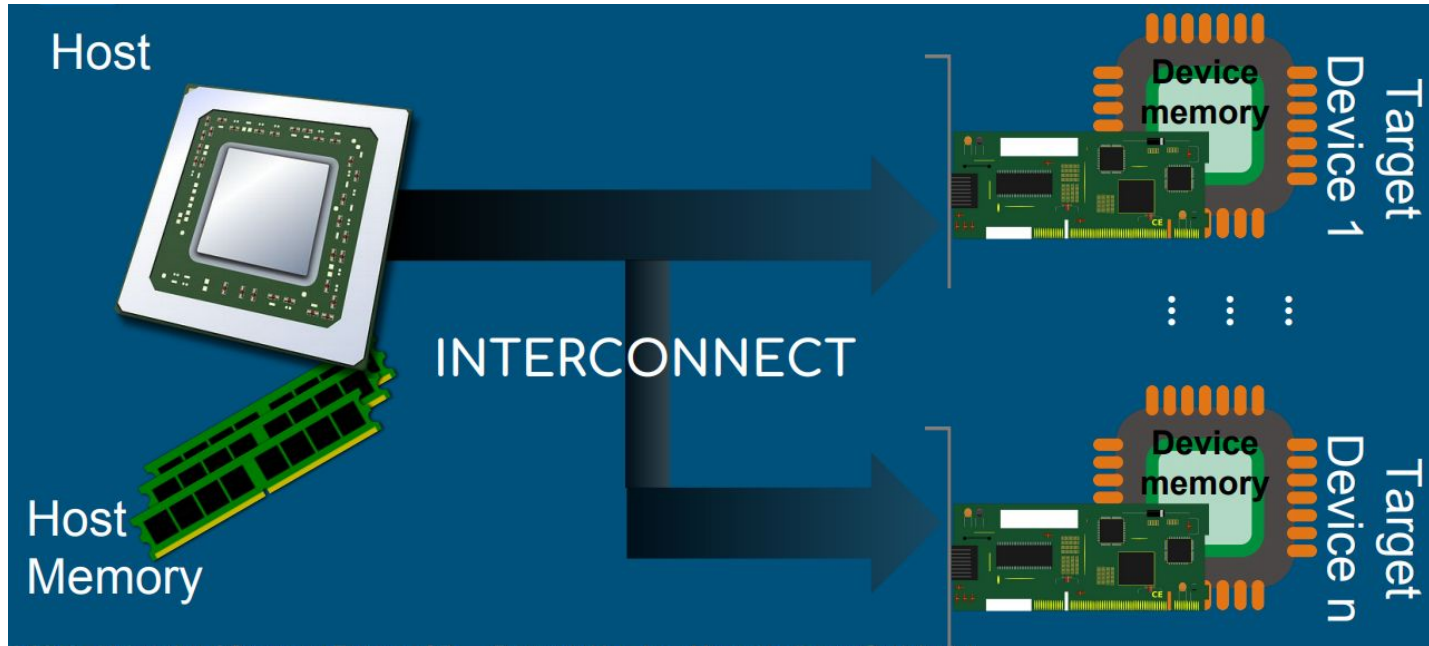


Problem

- The OpenMP specification is rapidly expanding
- Need offloading features to make use of accelerator devices
- OpenMP depends on compilers to implement its features, so new features are only usable once a compiler supports it
- Users follow the specification to learn usage, not implementation details



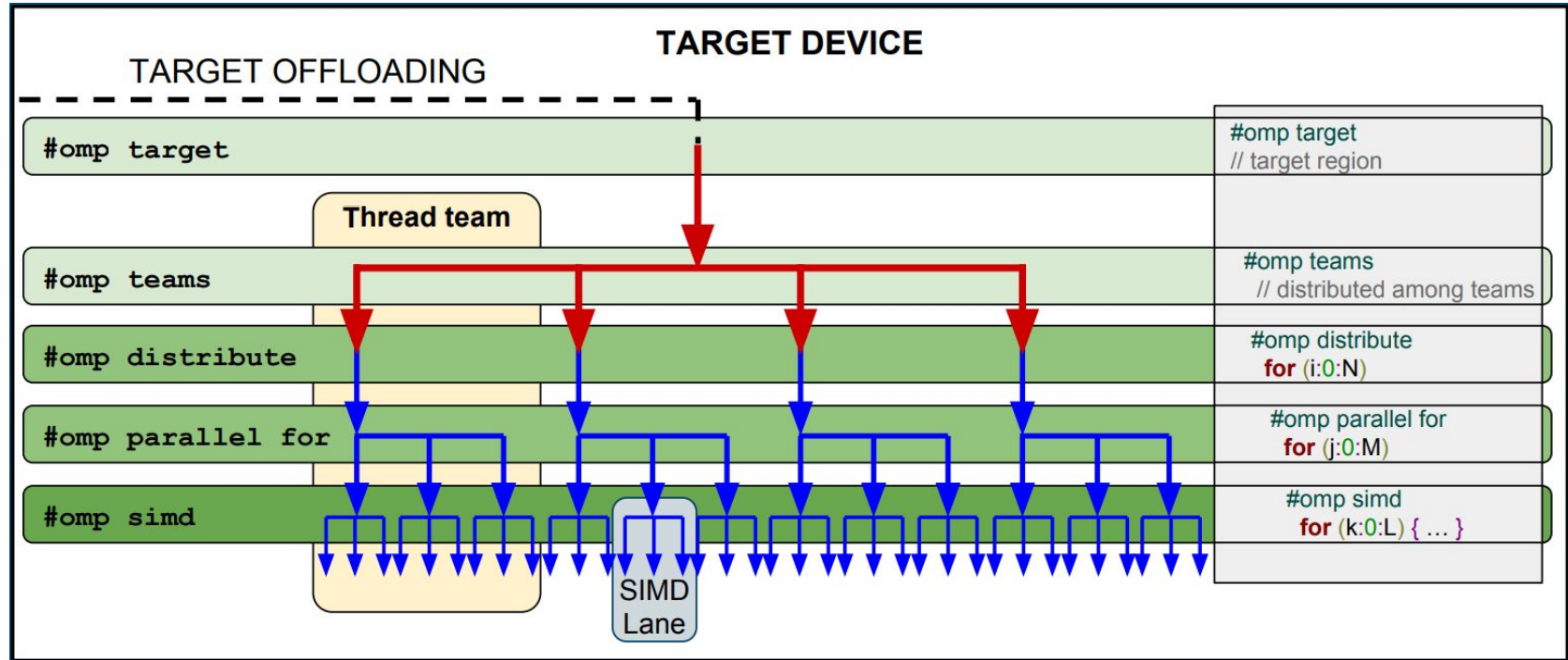
OpenMP 4.5 Data Environments



“Evaluating Support for OpenMP Offload Features,” Jose Monsalve Diaz



OpenMP 4.5 Execution Model



"Evaluating Support for OpenMP Offload Features," Jose Monsalve Diaz




target teams distribute construct

```
#pragma omp target teams distribute map(tofrom: a[0:1024])  
for (int i = 0; i < 1024; ++i) {  
    A[i] *= A[i];  
}
```



Target teams distribute construct

Target: map variables to device and execute construct on the device




```
#pragma omp target teams distribute map(tofrom: a[0:1024])  
for (int i = 0; i < 1024; ++i) {  
    A[i] *= A[i];  
}
```



Target teams distribute construct

Teams: create a league of thread teams




```
#pragma omp target teams distribute map(tofrom: a[0:1024])
for (int i = 0; i < 1024; ++i) {
    A[i] *= A[i];
}
```



Target teams distribute construct

Distribute: divide the loop iterations amongst the master threads of each team

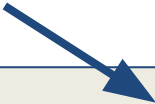


```
#pragma omp target teams distribute map(tofrom: a[0:1024])
for (int i = 0; i < 1024; ++i) {
    A[i] *= A[i];
}
```



Target teams distribute construct

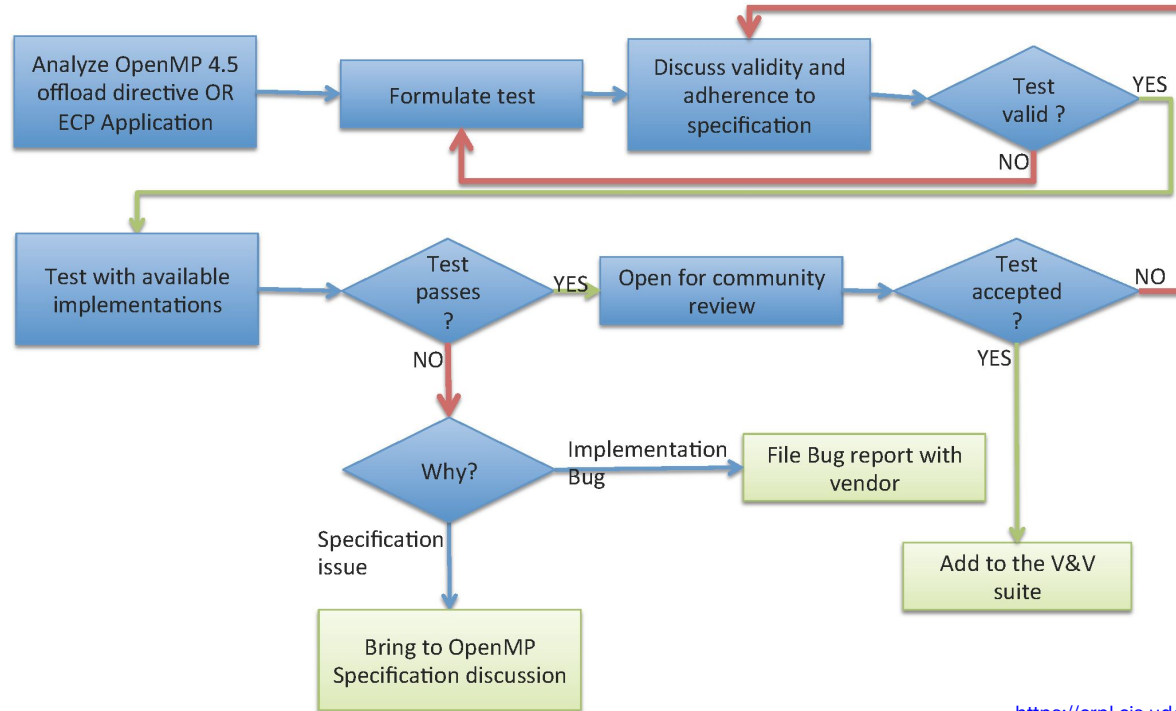
Map: specify mapping behavior for a list of variables



```
#pragma omp target teams distribute map(tofrom: a[0:1024])
for (int i = 0; i < 1024; ++i) {
    A[i] *= A[i];
}
```



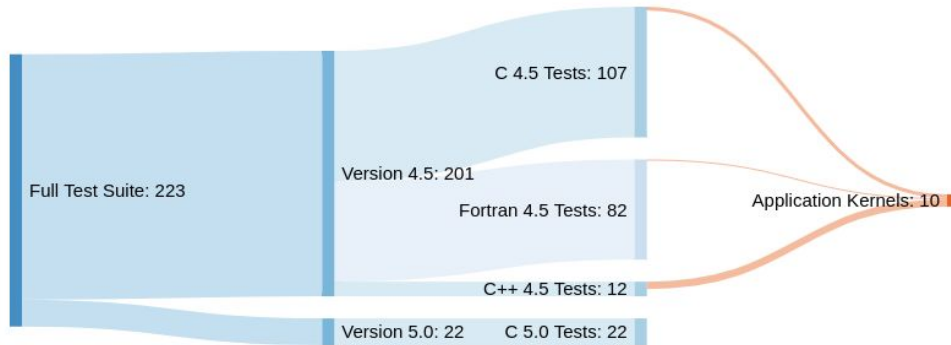
Workflow



<https://crpl.cis.udel.edu/ompvsolve/project/workflow/>



State of the Suite



- 223 individual tests in the suite
- 21 bugs filed with vendors
- All clauses in 4.5 spec are covered for these constructs:
 - target
 - target teams distribute
 - target teams distribute parallel for
 - target data
 - target enter/exit data
 - target update
- 4.5 Coverage in progress for:
 - task
 - target simd

<http://sankeymatic.com/>

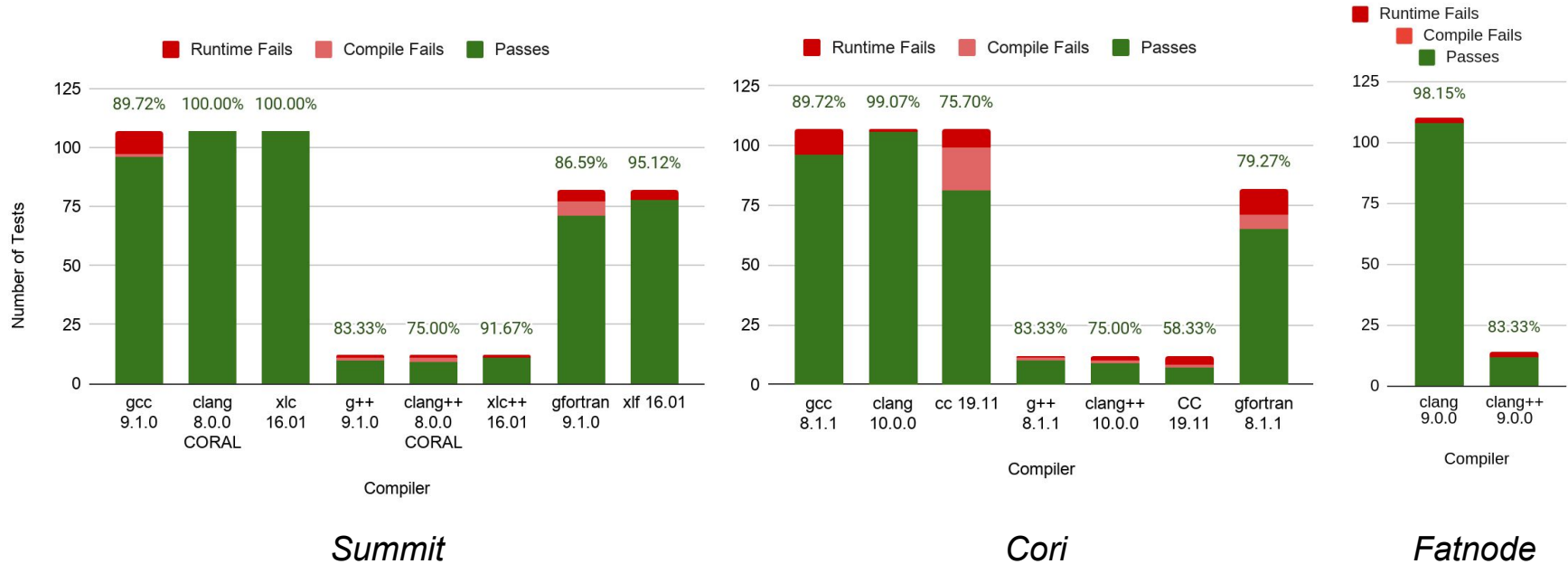


Test Environment

System	Summit	NERSC Cori (GPU nodes)	Fatnode
Model	IBM AC922	Cray XC40	Intel Xeon
Processors (per node)	IBM POWER9 x2	Intel Xeon Gold 6148 “Skylake” x2	Intel Xeon E5-2670 x1
Cores (per node)	42	40	8
Threads (per node)	168	80	16
Memory (per node)	512 GB	384 GB	384GB
Accelerator (per node)	NVIDIA V100 x6	NVIDIA V100 x8	NVIDIA K40 x 2
Compilers	GCC 9.1.0, XLC 16.01, Clang 8.0.0 CORAL	GCC 8.3.0, CCE 9.1.0 (CDT 19.11), Clang 10.0.0	GCC 9.0.1, Clang 9.0.1



4.5 Results (as of 13 April 2020)



5.0 Compiler Support



CRAY

- Cray
 - Two features: **depend** on **taskwait**, **OMP_DISPLAY_AFFINITY**



- GNU
 - Offers initial OpenMP 5.0 support (C/C++ only)



- Intel
 - Standard C/C++ compiler does not support target
 - Requires separate toolkit, supports intel hardware (integrated graphics only)



- Clang
 - In development:
<https://clang.llvm.org/docs/OpenMPSupport.html#openmp-implementation-details>



Performance Metrics

- Would be most reliable on local systems
 - Timer
 - Memory reads & writes
 - Heap & cache

```
int main() {  
  
    OMPVV_START();  
  
    int errors = 0;  
  
    int is_offloading;  
    OMPVV_TEST_AND_SET_OFFLOADING(is_offloading);  
  
    OMPVV_TEST_AND_SET_VERBOSE(errors, test_map_to());  
    OMPVV_REPORT_AND_RETURN(errors);  
  
    OMPVV_STOP();  
}
```



Improving the visual report

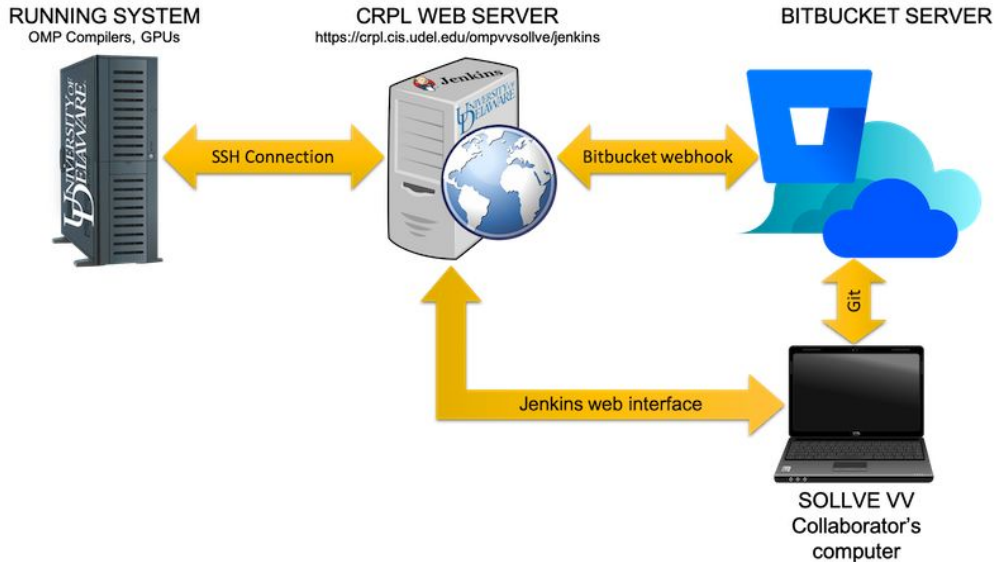
- Example:
https://crpl.cis.udel.edu/ompvvsolve/result_report/results.html?result_report=a189efb91
- Adding plots
 - Speed stats
 - Memory stats
- Adding warning information
 - Test may pass, but provide warnings such as “Offloading is not enabled, test may no longer be valid if not targeting a device”.



Continuous Integration

Architecture

Following is a depiction of the Jenkins infrastructure



- Ensure testsuite remains stable
- Confirm there are no issues with compilers (is this relevant?)
- Not possible with Summit



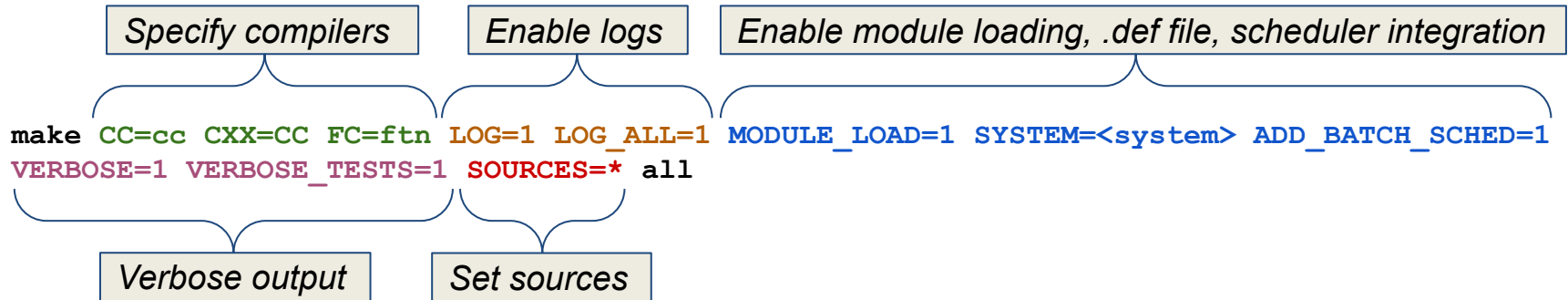
Other Changes

- Device Discovery
 - Make the suite more device-friendly
 - Remove headache of setting up new system (.def)
- Improve test coverage for C++
 - Possibly split the header file into a C and C++ version



Obtaining and Running the Suite

- First, clone the repo: https://github.com/SOLLVE/sollve_vv
- Set up `<system>.def` file and `make.def` according to your system
- Compile and run tests (after obtaining interactive job):



Reporting Results

- When `LOG=1` is provided, `make` will create a log folder with logs of test results
- Several recipes to view results:
 - `make report_summary` will give a short in-console overview of results
 - `make report_json` and `make report_csv` will give formatted data useful for post-processing
 - `make report_html` gives a user-friendly formatted page of results
 - **New/beta feature:** `make report_online` will upload the html report to the CRPL server so it can be easily viewed after running with the generated link!
 - Ex:
https://crpl.cis.udel.edu/ompvvsolve/result_report/results.html?result_report=a189efb91



Conclusions

- Knowing that support for many 5.0 features is unavailable, we must be cautious when interpreting the specification and writing tests which may not be immediately testable.
- Beyond expanding the suite and improving features for usage, new directions include possible performance measurement
- The V&V suite is now being used for regression tests in Cray, Intel, and AMD's development
- Website: <https://crpl.cis.udel.edu/ompvvsolve/>

