OpenMP 4.5 Validation and Verification Suite

Sunita Chandrasekaran (schandra@udel.edu)
Jose Diaz (josem@udel.edu),
ORNL: Oscar Hernandez (oscar@ornl.gov), Swaroop Pophale (pophaless@ornl.gov) or David Berndholt (david.bernholdt@gmail.com)

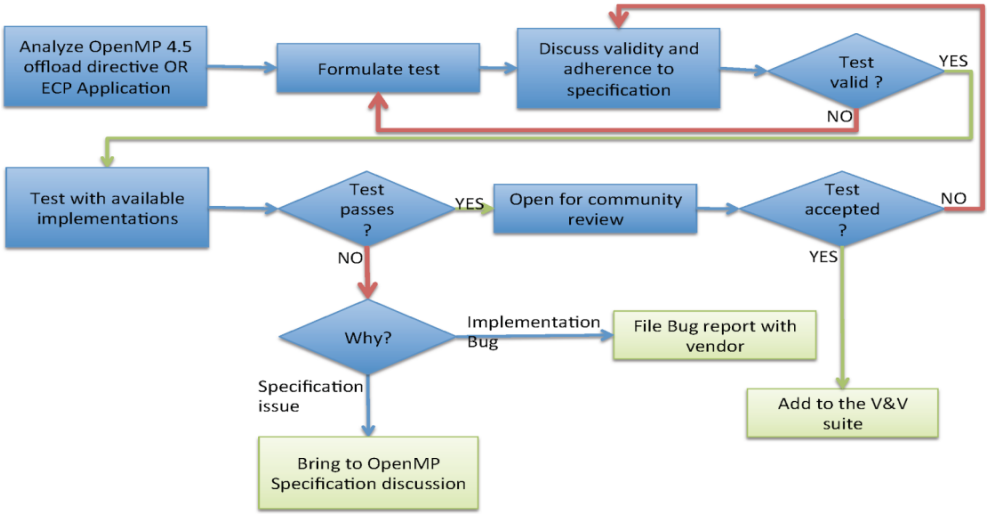
OpenMP Booth Wednesday 3.15PM
NOVEMBER 15, 2017

OpenMP 4.5 & beyond: Validation & Verification Testsuite project

- Evaluating functionality of the target offload across compilers & architectures
- Going forward: will cover the rest of the spec
- Part of the ECP SOLLVE Project
- In collaboration with ORNL, Argonne, Brookhaven, LLNL, LLVM, Cray, GCC
- Currently working closely with the LLVM Community
- Keen to work more closely with the OpenMP org. to make a larger impact
- Mail any of us for access to https://bitbucket.org/crpl_cisc/sollve_vv

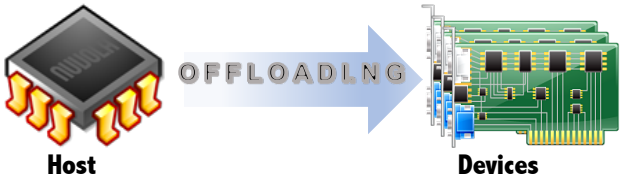
CONTACT *ORNL: Oscar Hernandez (oscar@ornl.gov), Swaroop Pophale (pophaless@ornl.gov)*
UDEL: Sunita Chandrasekaran (schandra@udel.edu), Jose Diaz (josem@udel.edu)

Testsuite Project Workflow



OpenMP 4.5 & beyond: Validation & Verification Testsuite project

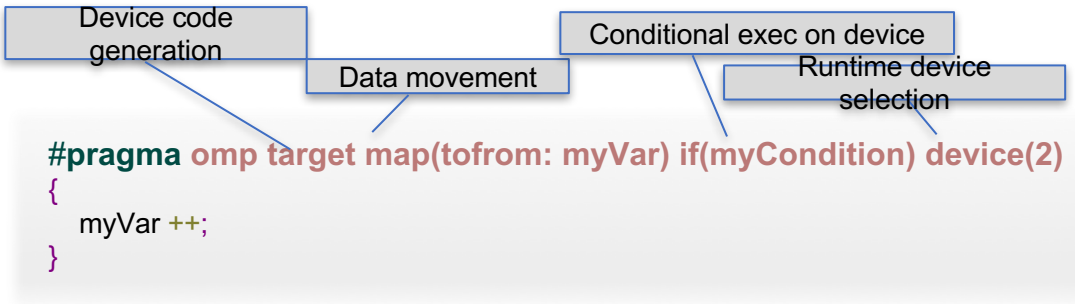
- Focusing heavily on Target offloading model
- Creating functional and unit tests
- Target platforms include ORNL's TITAN and SummitDev (representative exascale systems)
- Collaboration with ECP application developers to create representative scientific use cases



CONTACT *ORNL: Oscar Hernandez (oscar@ornl.gov), Swaroop Pophale (pophaless@ornl.gov)*
UDEL: Sunita Chandrasekaran (schandra@udel.edu), Jose Diaz (josem@udel.edu)

Target offloading

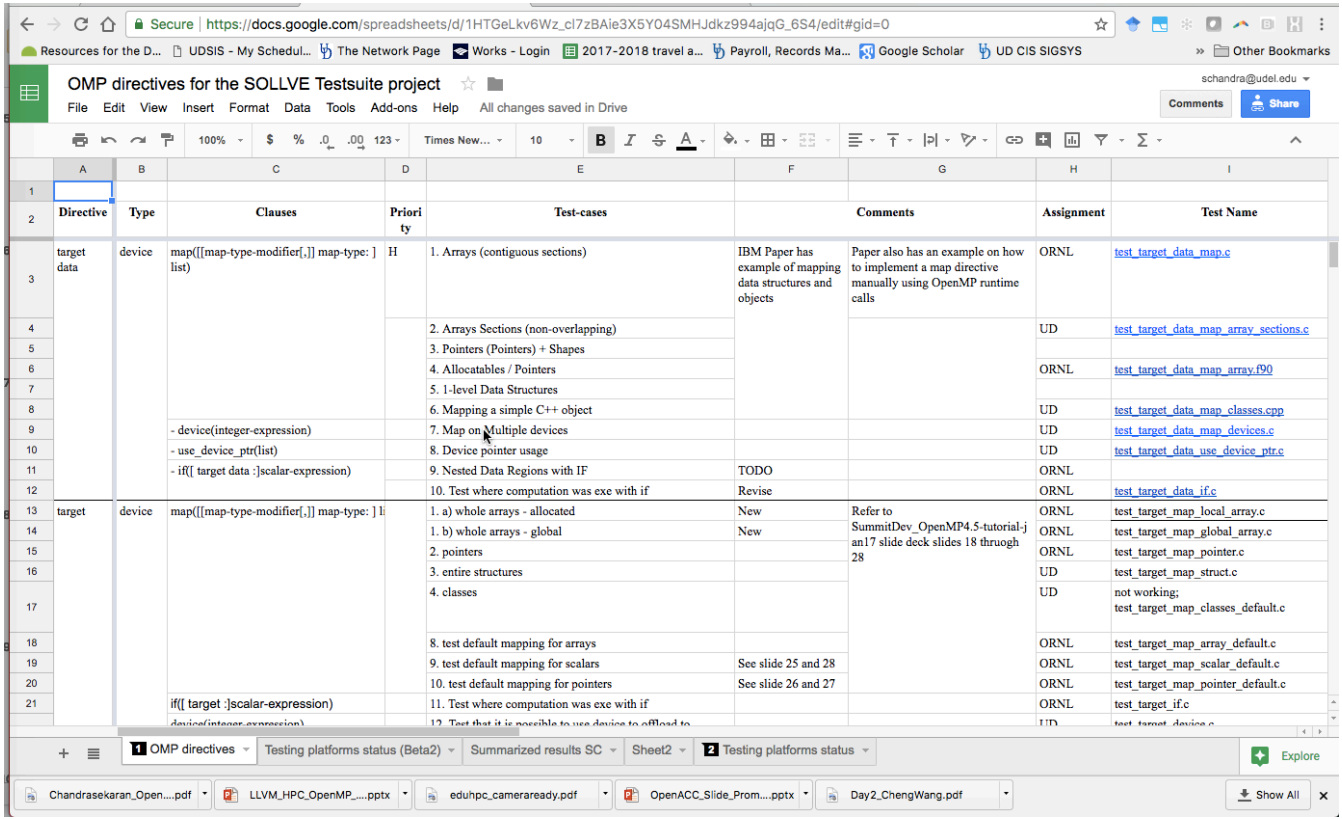
- **Host centric execution of code:** Offloading directives provides the compiler with hints to create device executable code, as well as inline all the necessary calls for device initialization, code execution and data movement between host and device.
- OpenMP frees the programmer from bookkeeping data allocation and movement, as well as separate compilation of code for host and device.
- OpenMP 4.5 in particular provides more control to the programmer to handle data movement between host and device.



Current Status

- Implemented 20+ V&V test cases for structured data map directives and runtime library calls.
- Committed patches to the LLVM compiler community on the V&V and a collaboration was established with the University of Delaware, ANL, BNL and LLNL.
- Created wiki pages which detail how to use the V&V repository and how to use llvm-lit infrastructure of LLVM to run the tests.
- Increased the range of different compilers to test the suite (IBM XL compilers, LLVM, GCC, Cray)
- Discussed with the OpenMP ARB's examples committee how to interact with the V&V.

Directives, clauses and test cases



Secure | https://docs.google.com/spreadsheets/d/1HTGeLkv6Wz_cl7zBAie3X5Y04SMHJdkz994ajqG_6S4/edit#gid=0

Resources for the D... | UDSIS - My Schedul... | The Network Page | Works - Login | 2017-2018 travel a... | Payroll, Records Ma... | Google Scholar | UD CIS SIGSYS | Other Bookmarks

OMP directives for the SOLLVE Testsuite project

schandra@udel.edu

File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

Comments | Share

	A	B	C	D	E	F	G	H	I
1									
2	**Directive**	**Type**	**Clauses**	**Priority**	**Test-cases**	**Comments**		**Assignment**	**Test Name**
3	target data	device	map([[map-type-modifier[,]] map-type:] list)	H	1. Arrays (contiguous sections)	IBM Paper has example of mapping data structures and objects	Paper also has an example on how to implement a map directive manually using OpenMP runtime calls	ORNL	test_target_data_map.c
4					2. Arrays Sections (non-overlapping)			UD	test_target_data_map_array_sections.c
5					3. Pointers (Pointers) + Shapes				
6					4. Allocatables / Pointers			ORNL	test_target_data_map_array.f90
7					5. 1-level Data Structures				
8					6. Mapping a simple C++ object			UD	test_target_data_map_classes.cpp
9			- device(integer-expression)		7. Map on Multiple devices			UD	test_target_data_map_devices.c
10			- use_device_ptr(list)		8. Device pointer usage			UD	test_target_data_use_device_ptr.c
11			- if([target data :]scalar-expression)		9. Nested Data Regions with IF	TODO		ORNL	
12					10. Test where computation was exe with if	Revise		ORNL	test_target_data_if.c
13	target	device	map([[map-type-modifier[,]] map-type:] li		1. a) whole arrays - allocated	New	Refer to SummitDev_OpenMP4.5-tutorial-jan17 slide deck slides 18 thruogh 28	ORNL	test_target_map_local_array.c
14					1. b) whole arrays - global	New		ORNL	test_target_map_global_array.c
15					2. pointers			ORNL	test_target_map_pointer.c
16					3. entire structures			UD	test_target_map_struct.c
17					4. classes			UD	not working; test_target_map_classes_default.c
18					8. test default mapping for arrays			ORNL	test_target_map_array_default.c
19					9. test default mapping for scalars	See slide 25 and 28		ORNL	test_target_map_scalar_default.c
20					10. test default mapping for pointers	See slide 26 and 27		ORNL	test_target_map_pointer_default.c
21			if([target :]scalar-expression)		11. Test where computation was exe with if			ORNL	test_target_if.c

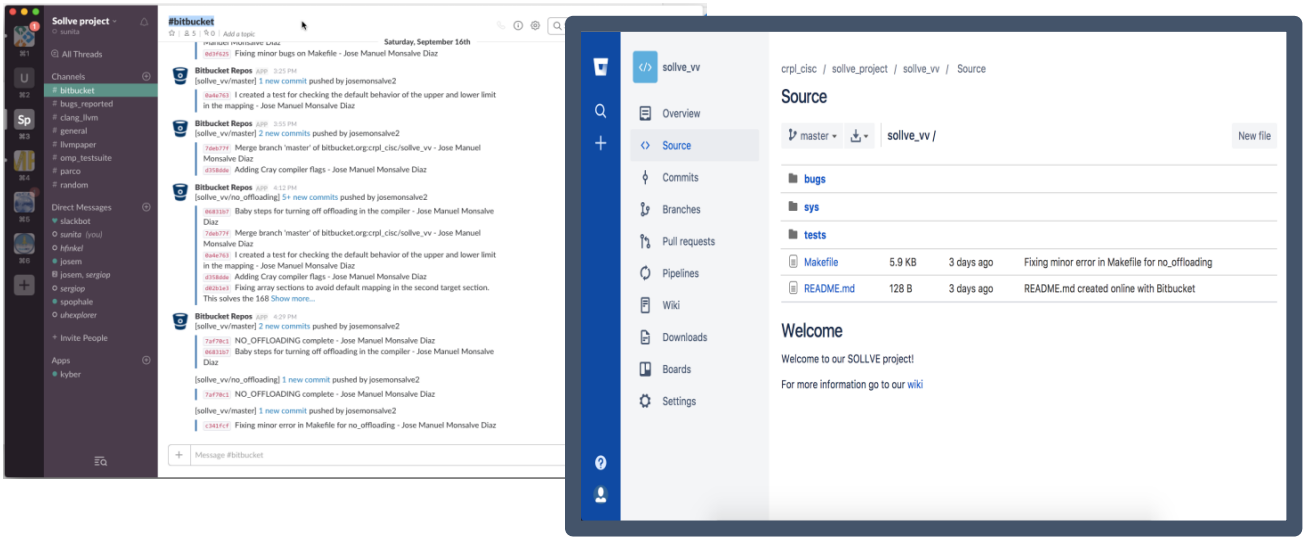
1 OMP directives | Testing platforms status (Beta2) | Summarized results SC | Sheet2 | 2 Testing platforms status | Explore

Chandrasekaran_Open....pdf | LLVM_HPC_OpenMP_....pptx | eduhpc_cameraready.pdf | OpenACC_Slide_Prom....pptx | Day2_ChengWang.pdf | Show All

Offloading to multiple devices

- Distributes each row of a matrix to exactly one of the available devices
- Line 8 – without the map clause map(from:h_matrix[dev*N:N]) the test was reporting success
  - But at runtime, the entire h_matrix was implicitly copied to each one of the devices.
  - Each device had a copy of the h_matrix array instead of only a portion of the array (a row) per device
- Line 8 – added h_matrix array present implicitly in a target region is not larger than the original size already mapped in the outer region by the target data construct in line 7
- It is important to clarify that the additional map(from: h_matrix[dev ∗N:N]) will not perform the data movement operation since the h_matrix array was already mapped in the outer target data region.

```
1  int test_map_device() {
2    int num_dev = omp_get_num_devices(), sum[num_dev], errors = 0;
3    int* h_matrix = (int*) malloc(num_dev*N*sizeof(int));
4
5    for (int dev = 0; dev < num_dev; ++dev) {
6  #pragma omp target data map(from: h_matrix[dev*N:N]) device(dev)
7      {
8  #pragma omp target map(from: h_matrix[dev*N:N]) device(dev)
9        {
10         for (int i = 0; i < N; ++i)
11           h_matrix[dev*N + i] = dev;
12       } // end target
13     } // end target data
14   }
15
16   // checking results
17   errors = 0;
18   for (int dev = 0; dev < num_dev; ++dev) {
19     sum[dev] = h_matrix[dev*N + 0];
20     for (int i = 1; i < N; ++i)
21       sum[dev] += h_matrix[dev*N + i];
22     errors |= (dev * N != sum[dev]);
23   }
24
25   return errors;
26 }
```



Contact: ORNL: Oscar Hernandez(oscar@ornl.gov), Swaroop Pophale(pophaless@ornl.gov)
or
UDEL: Jose Diaz (josem@udel.edu), Sunita Chandrasekaran (schandra@udel.edu)

RESULTS (so far)

Test	Lang	Platform: Summitdev Compiler: Clang 3.8.0 ORNL Version	Platform: Summitdev Compiler: IBM XL 13.1.6	Platform: Titan Compiler: CCE 8.6.3	Platform: Summitdev Compiler: GCC 7.1.1
offloading_success.c	C	PASS	PASS	PASS	PASS
offloading_success.cpp	C++	PASS	PASS	PASS	PASS
target/test_target_is_device_ptr.c	C	PASS	PASS	PASS	PASS
target/test_target_map_array_default.c	C	PASS	PASS	PASS	PASS
target/test_target_map_pointer.c	C	PASS	PASS	PASS	PASS
target/test_target_map_pointer_default.c	C	PASS	PASS	PASS	PASS
target/test_target_map_scalar_default.c	C	PASS	PASS	PASS	PASS
target/test_target_if.c	C	PASS	PASS	PASS	PASS
target/test_target_map_global_arrays.c	C	PASS	PASS	PASS	PASS
target/test_target_map_local_array.c	C	PASS	PASS	PASS	PASS
target_data/test_target_data_if.c	C	PASS	PASS	PASS	PASS
target_data/test_target_data_map.c	C	PASS	PASS	PASS	PASS
target_enter_exit_data/test_target_enter_exit_data_async.c	C	PASS	PASS	PASS	PASS
target_enter_exit_data/test_target_enter_exit_data_map_global_array.c	C	PASS	PASS	PASS	PASS
target_enter_exit_data/test_target_enter_exit_data_map_malloced_array.c	C	PASS	PASS	PASS	PASS
target/test_target_map_struct_default.c	C	PASS	PASS	C. FAIL	PASS
target/test_target_device.c	C	PASS	PASS	C. FAIL	PASS
target/test_target_async.c	C	PASS	R. FAIL	PASS	PASS
target_data/test_target_data_map_array_sections.c	C	PASS	PASS	R. FAIL	PASS
target_data/test_target_data_map_classes.cpp	C++	PASS	PASS	C. FAIL	PASS
target_data/test_target_data_map_devices.c	C	PASS	PASS	C. FAIL	PASS
target_enter_exit_data/test_target_enter_exit_data_devices.c	C	PASS	PASS	C. FAIL	PASS
target_enter_exit_data/test_target_enter_exit_data_struct.c	C	PASS	PASS	C. FAIL	PASS
application_kernels/linked_list.c	C	PASS	PASS	C. FAIL	PASS
target_data/test_target_data_use_device_ptr.c	C	PASS	PASS	R. FAIL	C. FAIL
target/test_target_defaultmap.c	C	PASS	R. FAIL	R. FAIL	PASS
target/test_target_map_classes_default.cpp	C++	R. FAIL	R. FAIL	C. FAIL	C. FAIL
target_enter_exit_data/test_target_enter_exit_data_classes.cpp	C++	C. FAIL	R. FAIL	C. FAIL	C. FAIL

Next Steps

- Extract more test cases from applications use-cases to complement the ones written
- Identify performance-critical kernels from application
- Coordinate V&V work with the LLVM community – develop a schedule for the next batch of testcases
- V&V release for SC'17

Thank you !!

- Visit UDEL Booth #1043 to check out a poster on this topic for more updates
- Visit DOE Booth #613 on Wed 5pm for a demo on V&V