Evaluating Support for OpenMP Offload Features

Jose Monsalve Diaz
Swaroop Pophale
Kyle Friedline
Oscar Hernandez
David E. Bernholdt
Sunita Chandrasekaran

University of Delaware
ECP Exascale Computing Project
Oak Ridge National Laboratory

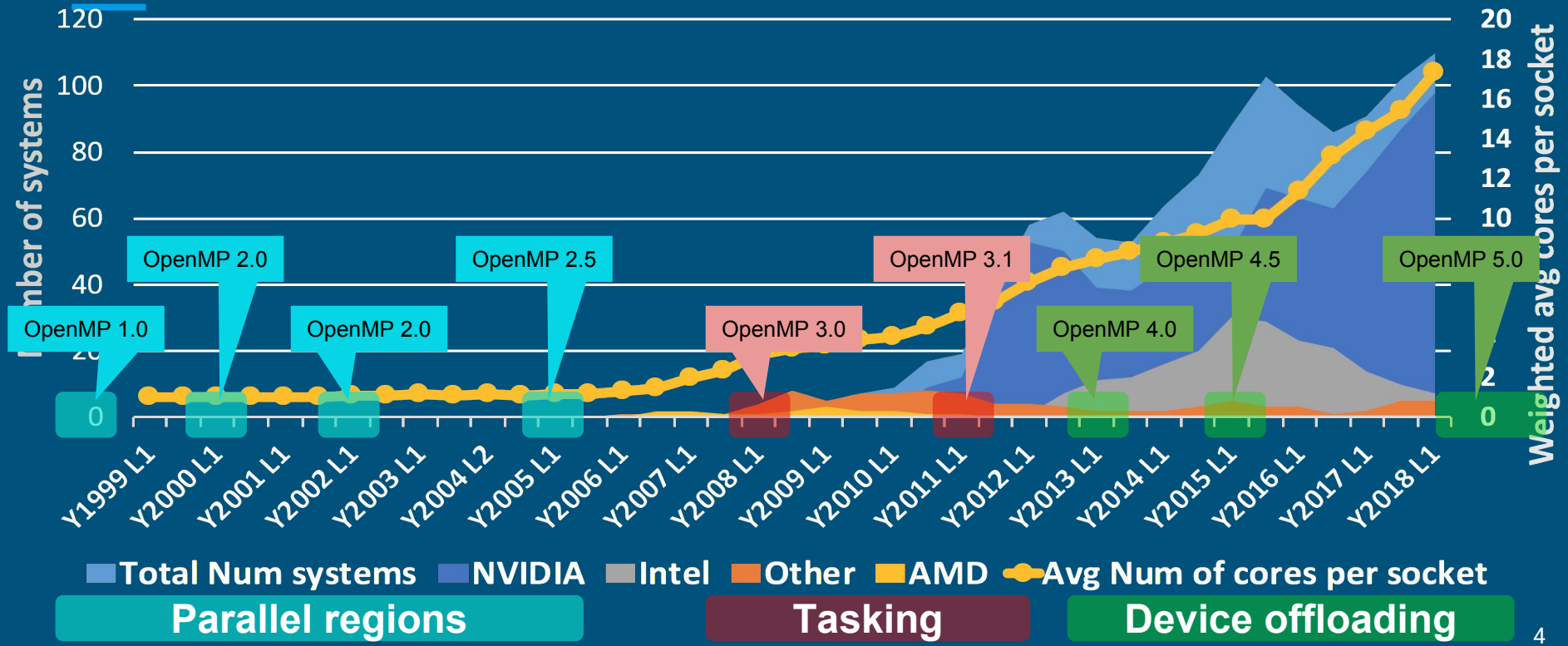
Outline

1. Introduction
2. Problem statement and contributions
3. OpenMP 4.5 offloading
4. Methodology
5. Experimental Setup
6. Results
7. Conclusions

Introduction

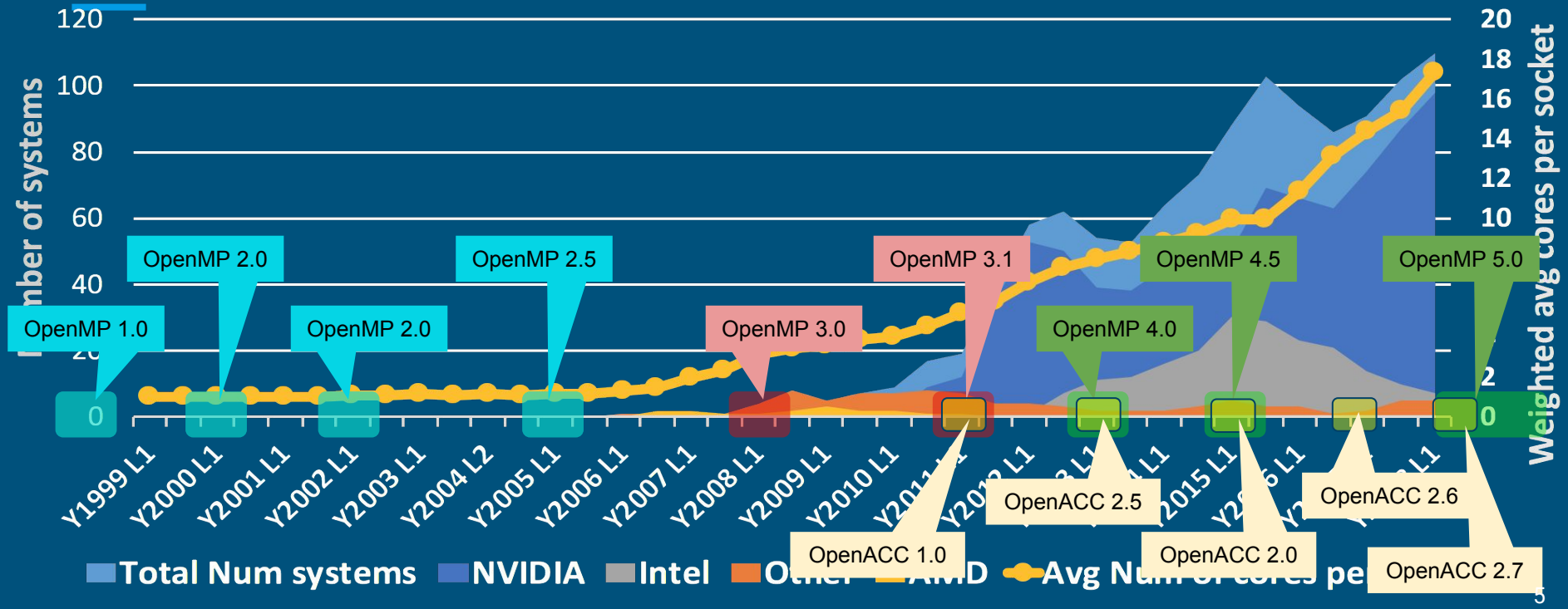
Current Trends in HPC

(Data from Top 500 list: www.top500.org)



Current Trends in HPC

(Data from Top 500 list: www.top500.org)



Why Accelerator devices?

- Power wall
- Simpler cores and considerably larger core count
- Programs taking advantage of SIMD and SPMD models
  - Better performance/Watt ratio
  - Benefit from large parallelism

But there are still programmability challenges...

Problem statement and contributions

Problem statement

Given the specifications of **OpenMP 4.5**, the multiple compiler implementations that exist, and the different systems where this programmer model will be used. **how to assess the level of compliance of implementations and systems with respect to the specifications document?**

Problem statement
Explanation

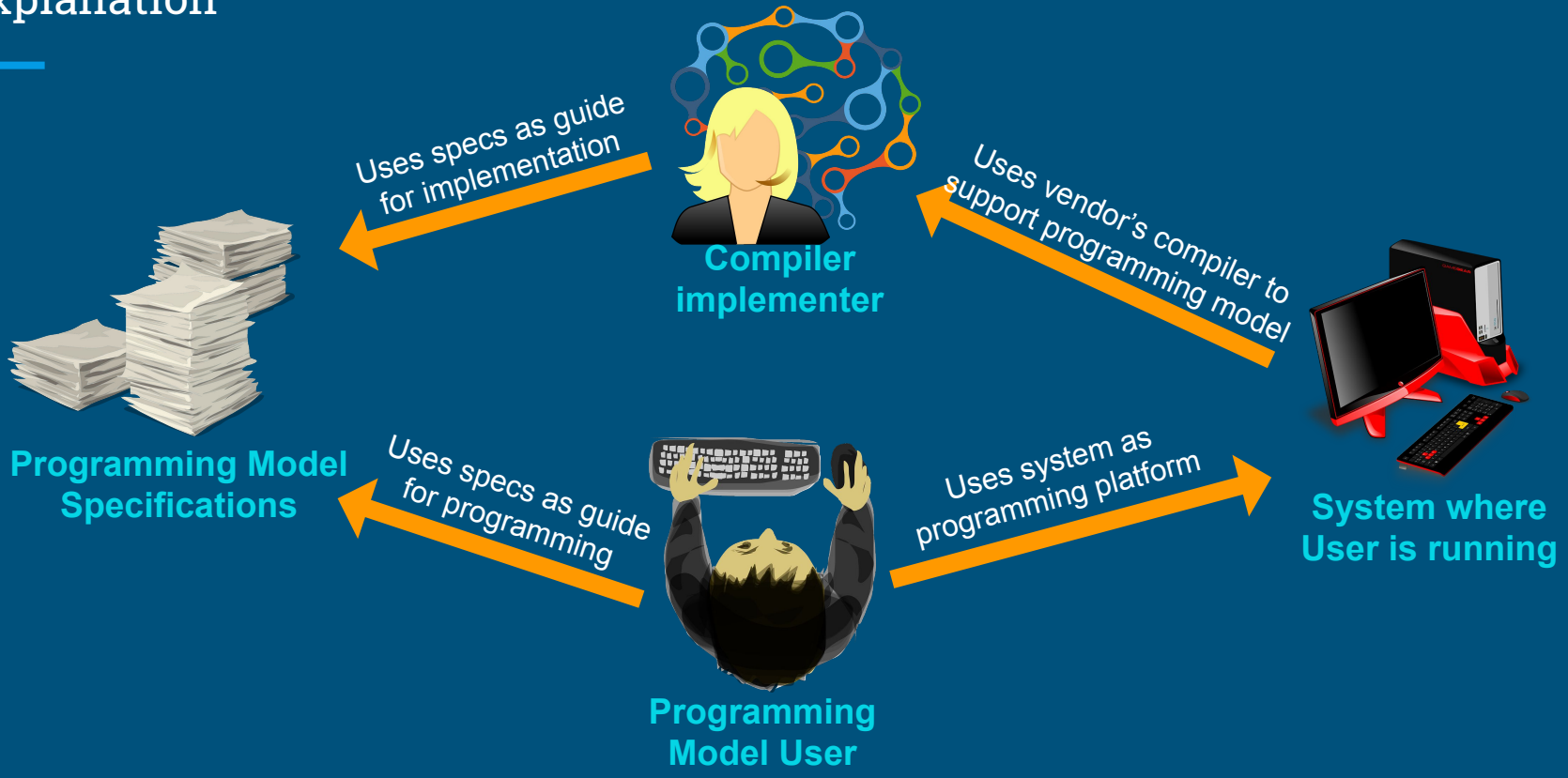
Programming model specifications document

"Legal" document that binds the implementation and the user

- Compiler developers guide their products based on the specifications documents. They must respect them to claim support
- Users do not need to learn implementation specific aspects of the programming model. They use the specifications
- What about the system that the user is running on?

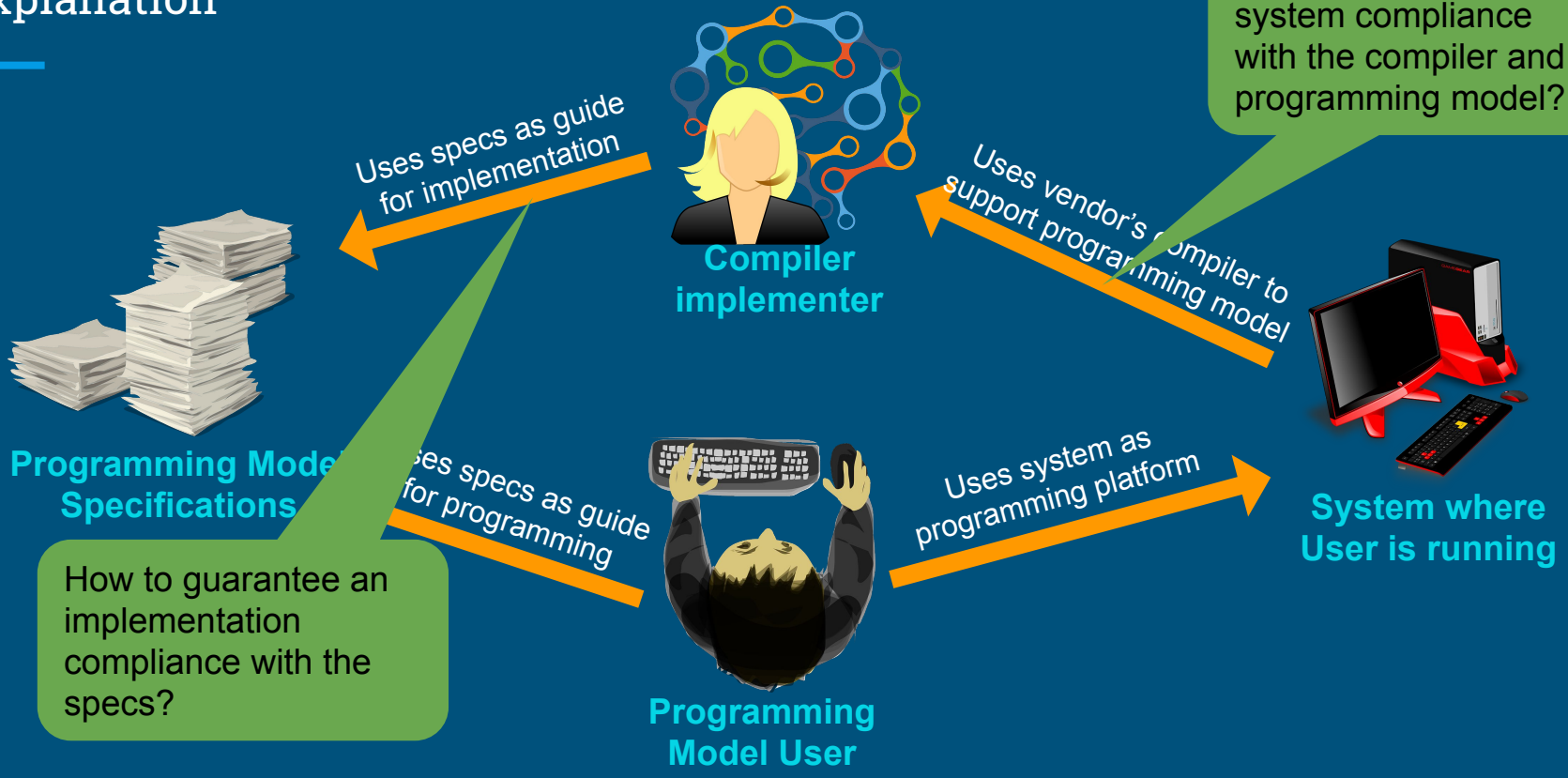
Problem statement

Explanation



Problem statement

Explanation



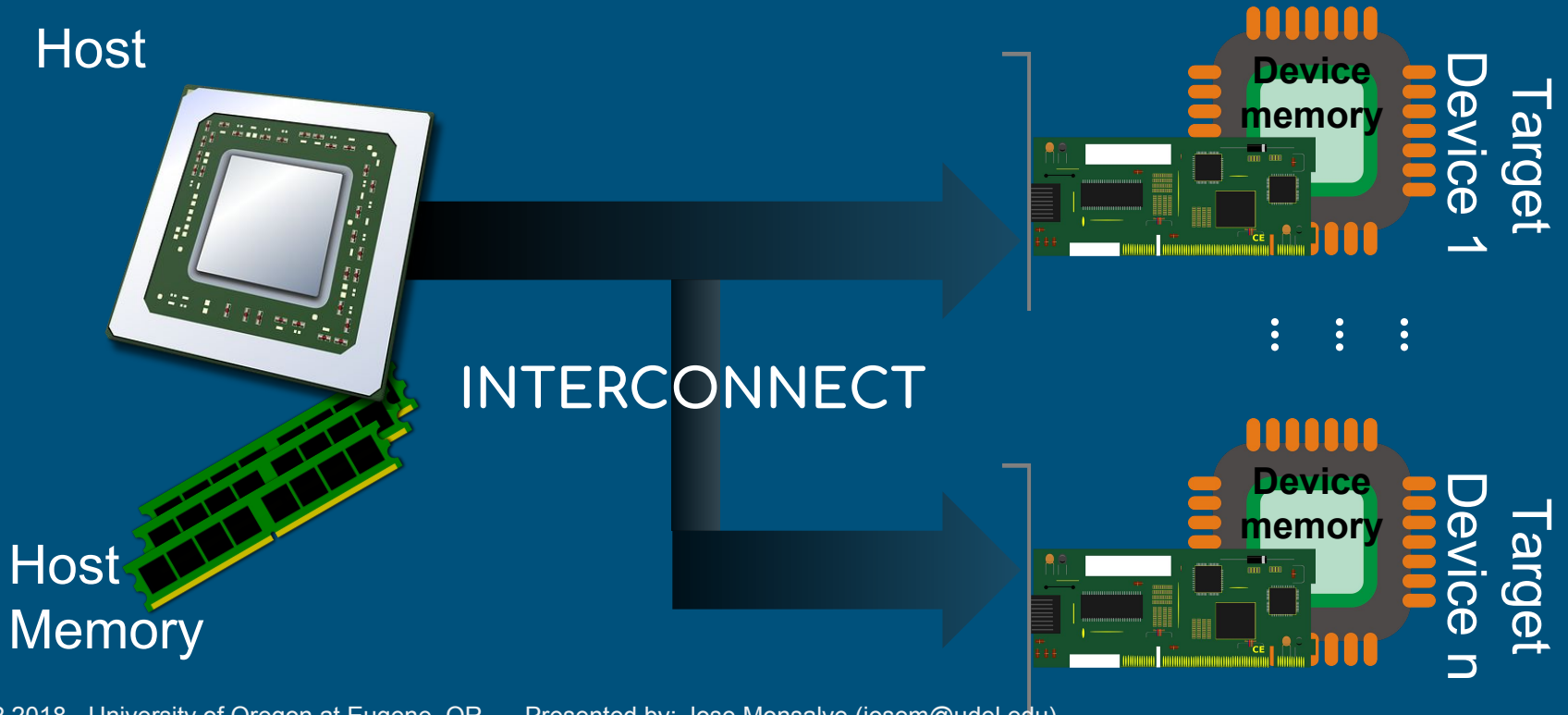
Contributions of this work

We are the “lawyers” of the OpenMP 4.5 Specifications:

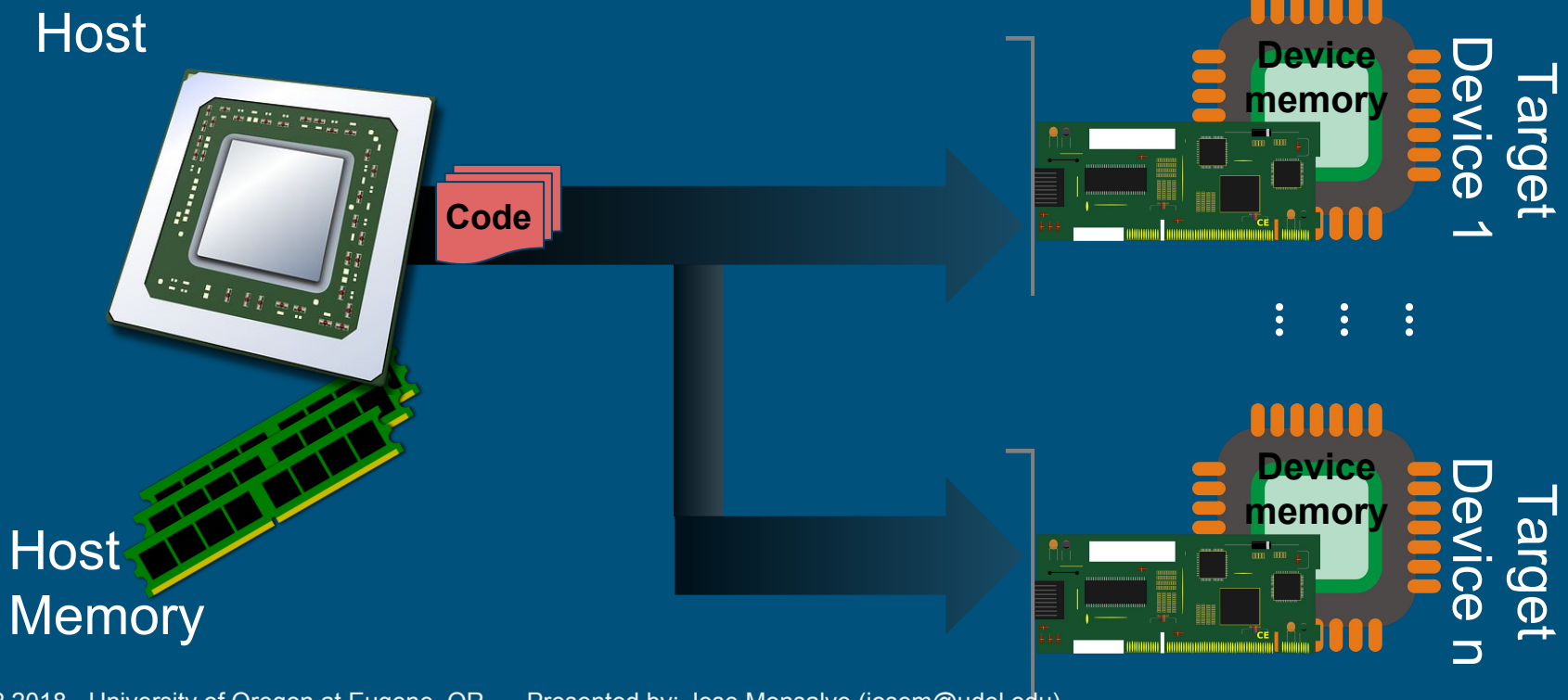
- Identify extent of OpenMP offload support (target directives) in OpenMP 4.5 implementations such as GCC, Clang/LLVM, IBM XL and Cray CCE
- Analyze support for common code kernels identified across a range of **DOE applications** and test their support across all accessible OpenMP 4.5 implementations
- Identify and report inconsistencies or bugs in specific implementations to their respective compiler developers
- Present performance data for different directives across different OpenMP 4.5 implementations

OpenMP 4.5 Offloading

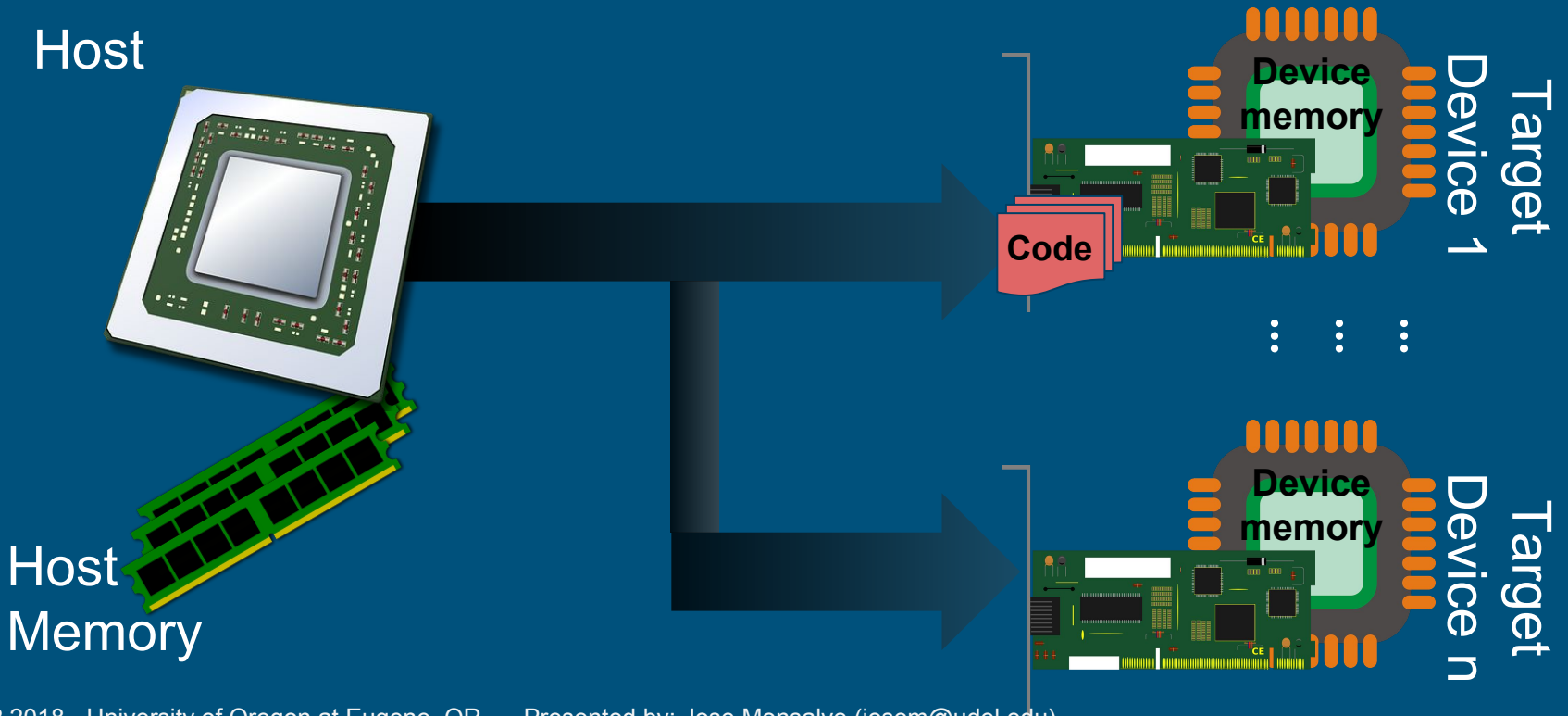
OpenMP 4.5 Machine model



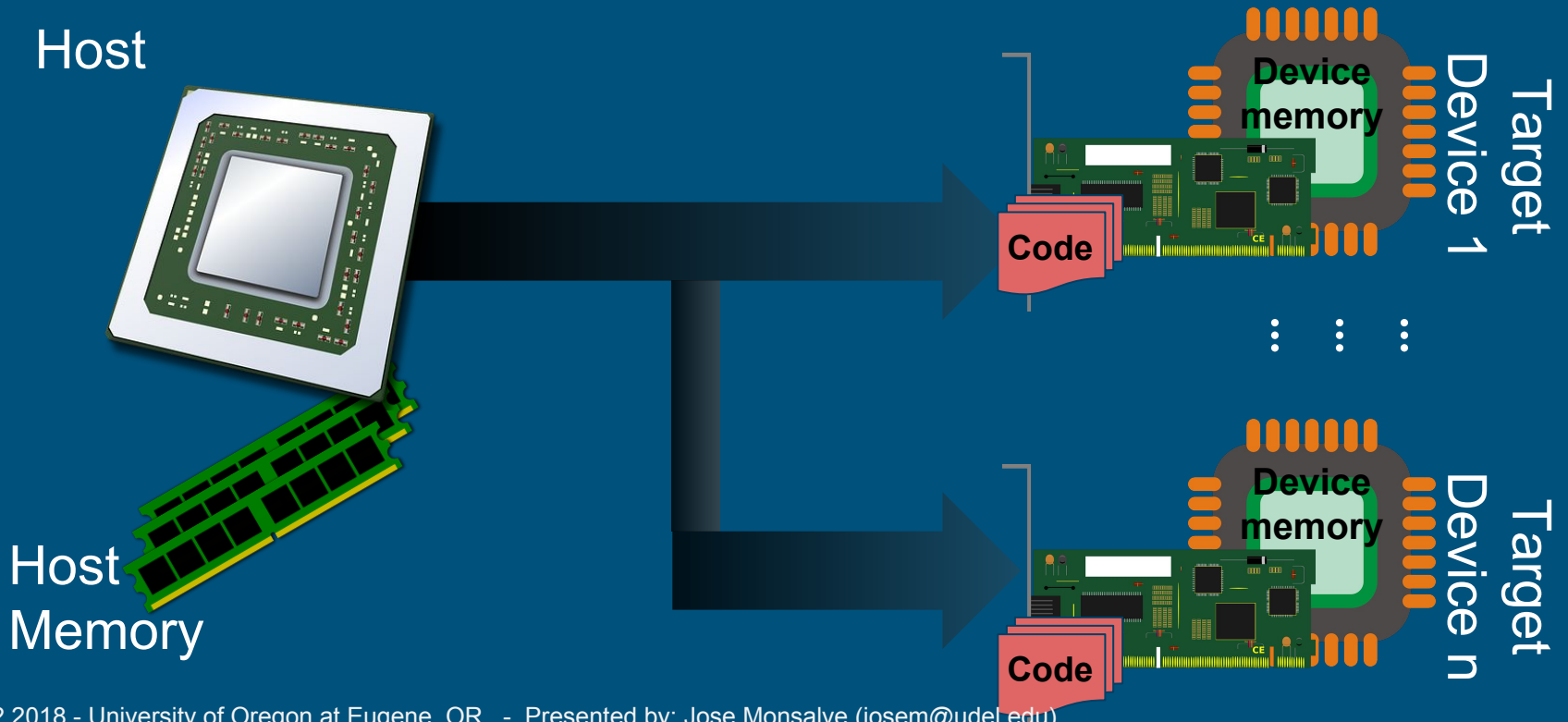
OpenMP 4.5 Offloading
Code Execution Model



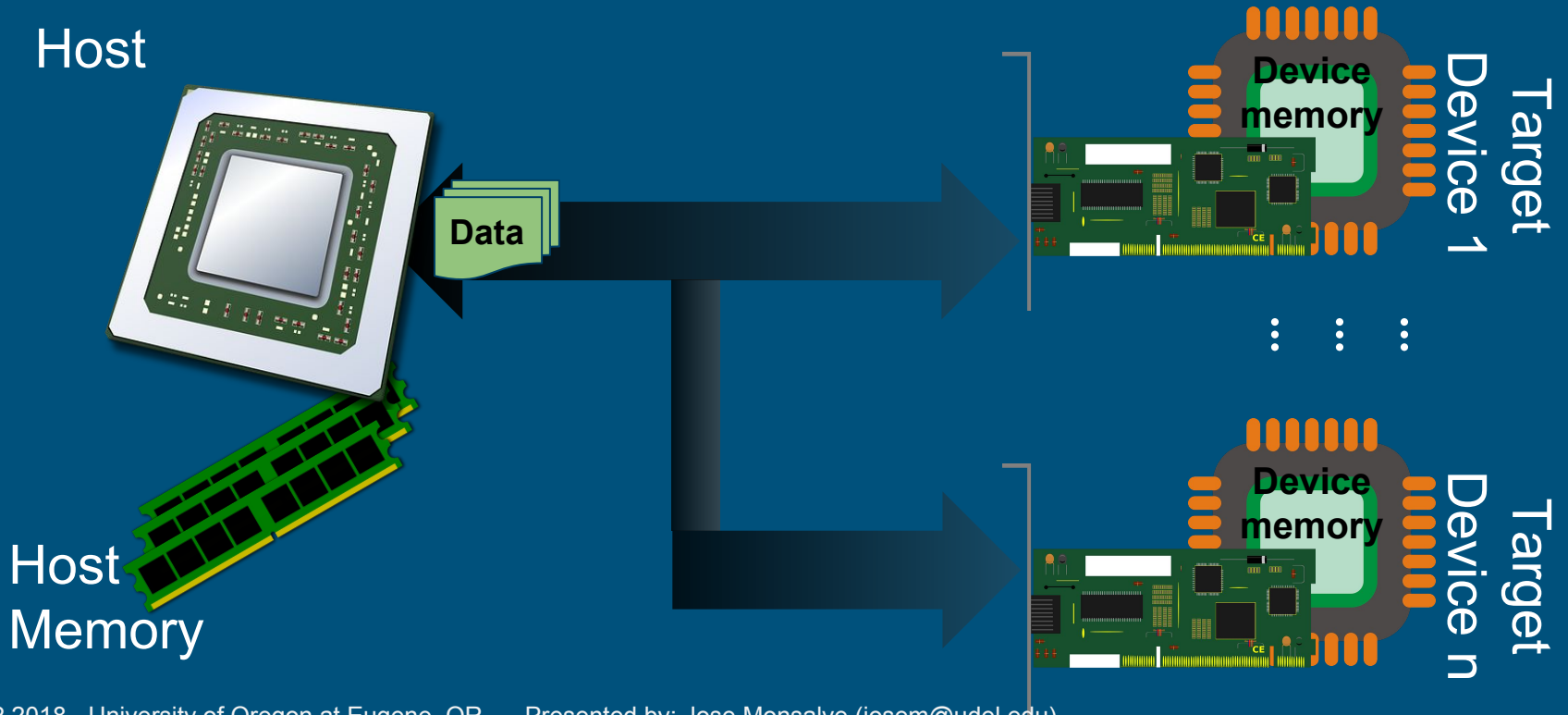
OpenMP 4.5 Offloading
Code Execution Model



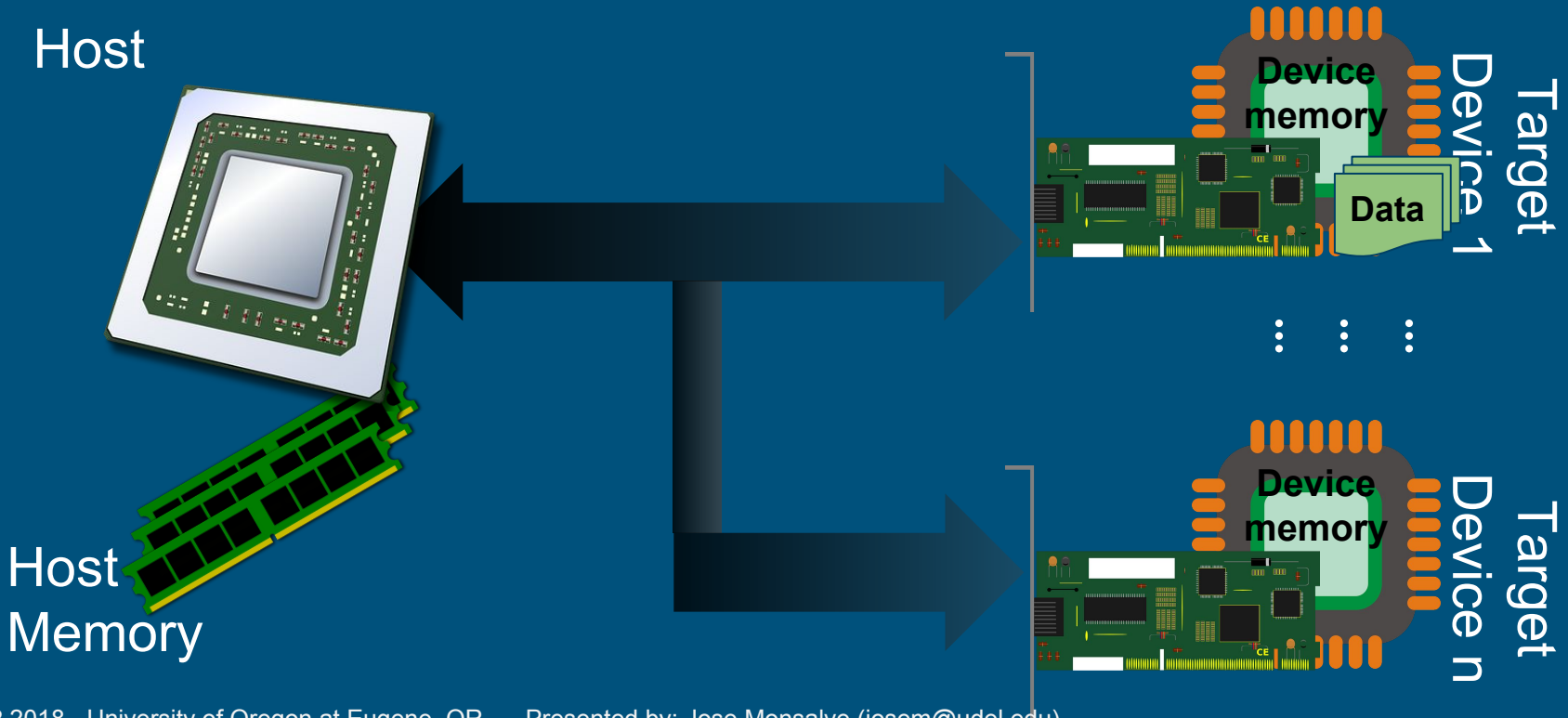
OpenMP 4.5 Offloading
Code Execution Model



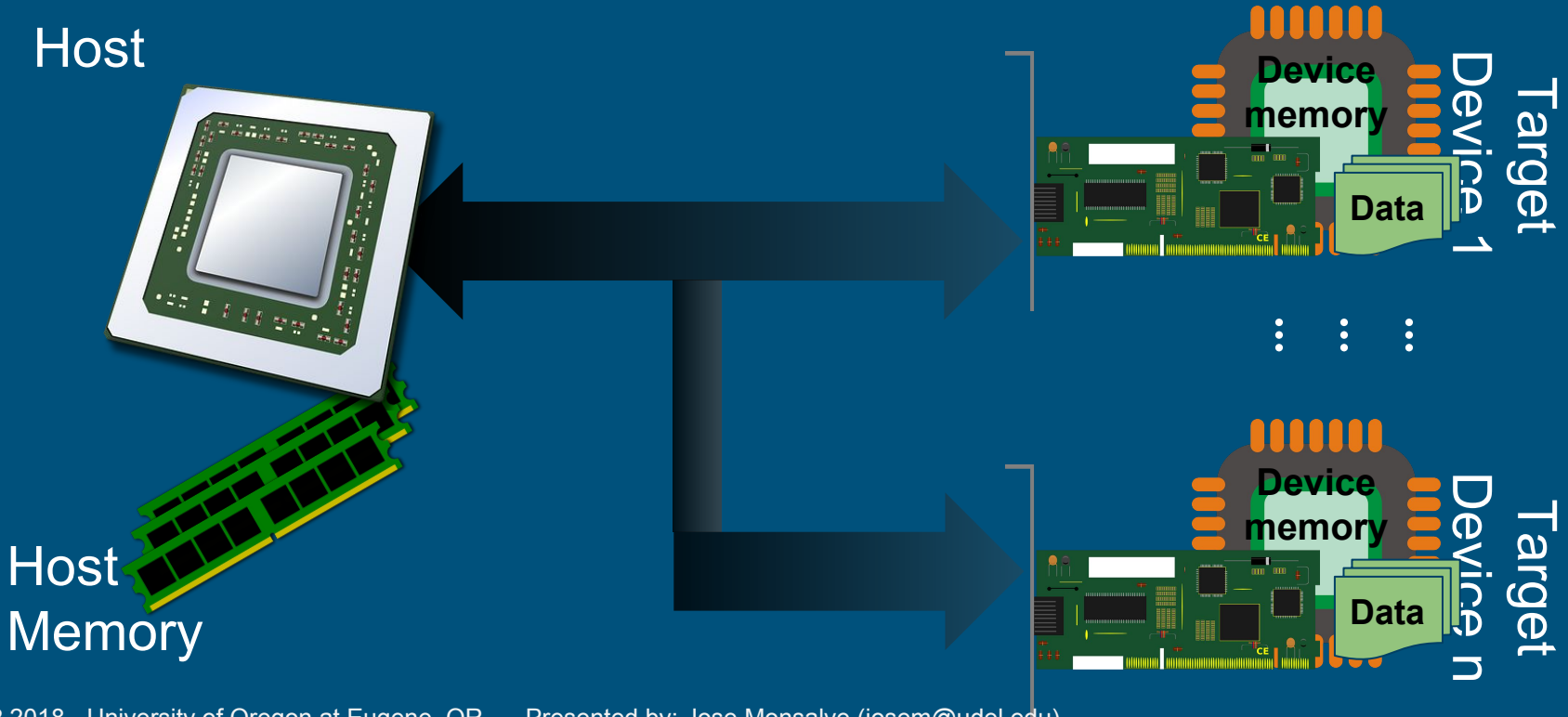
OpenMP 4.5 Offloading
Data management model



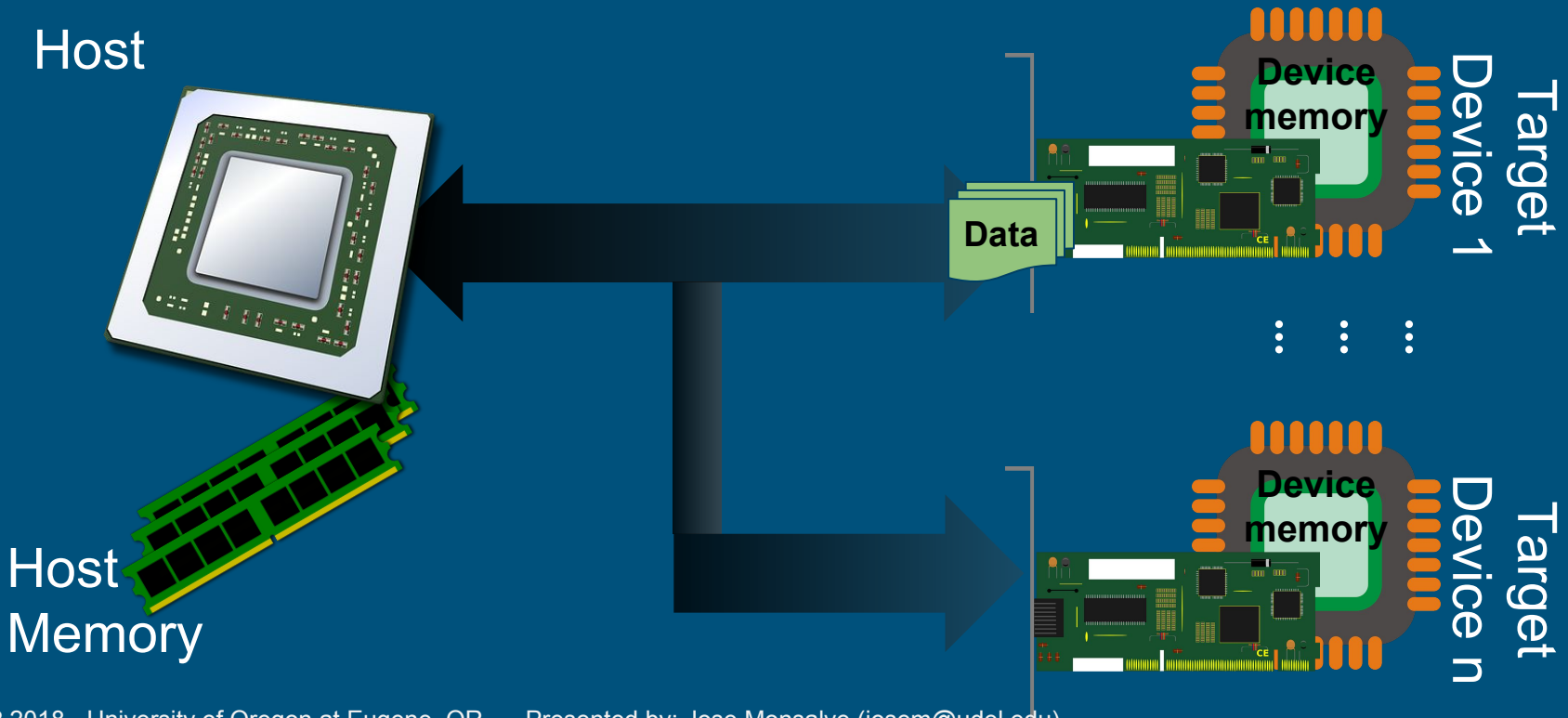
OpenMP 4.5 Offloading
Data management model



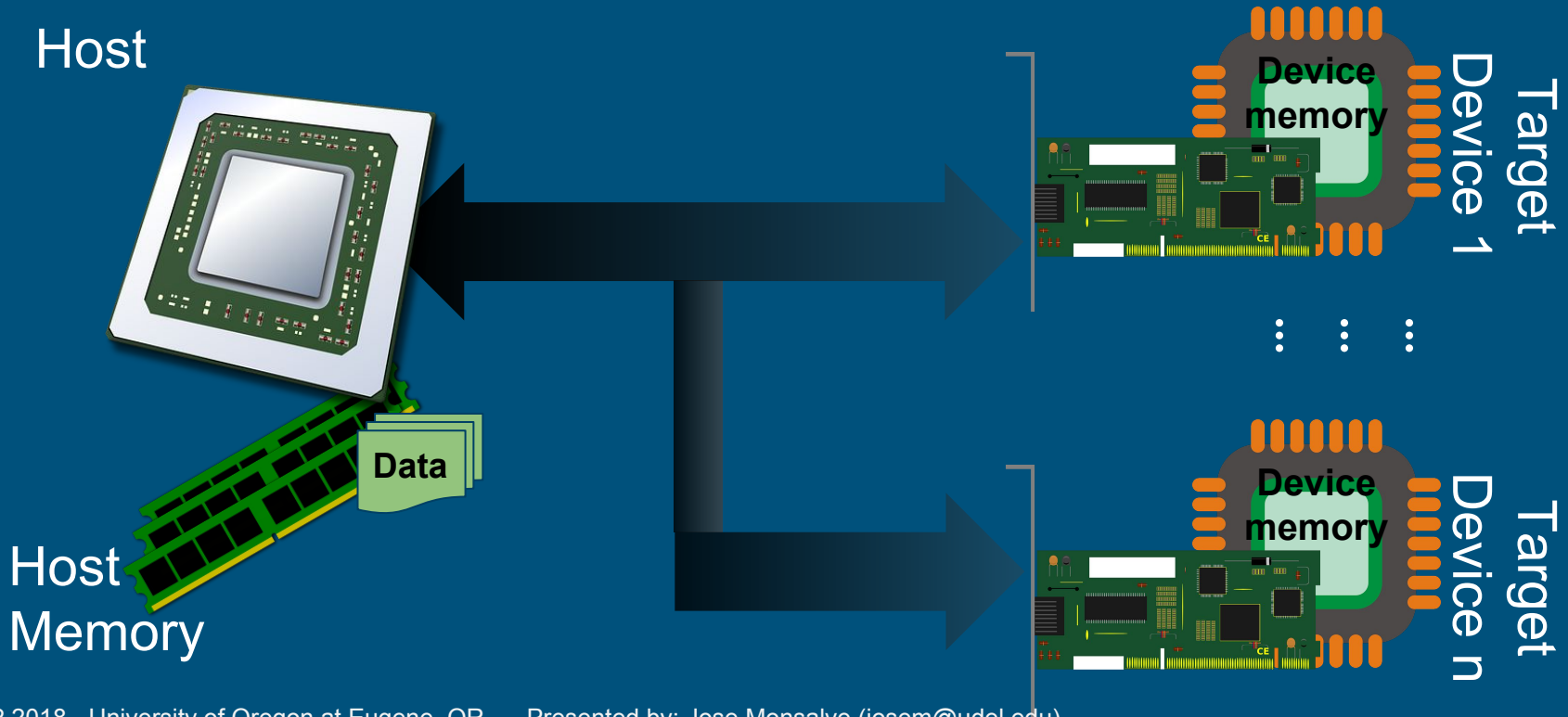
OpenMP 4.5 Offloading
Data management model



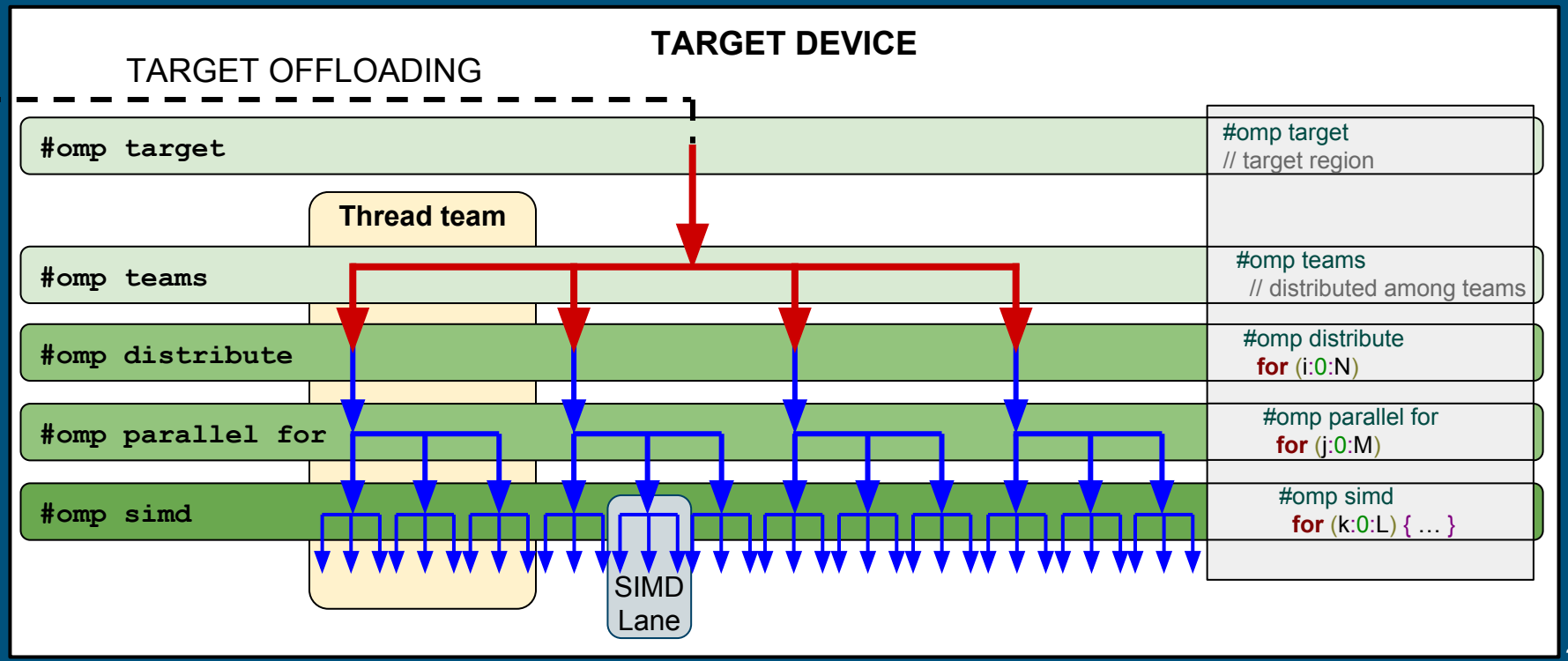
OpenMP 4.5 Offloading
Data management model



OpenMP 4.5 Offloading
Data management model



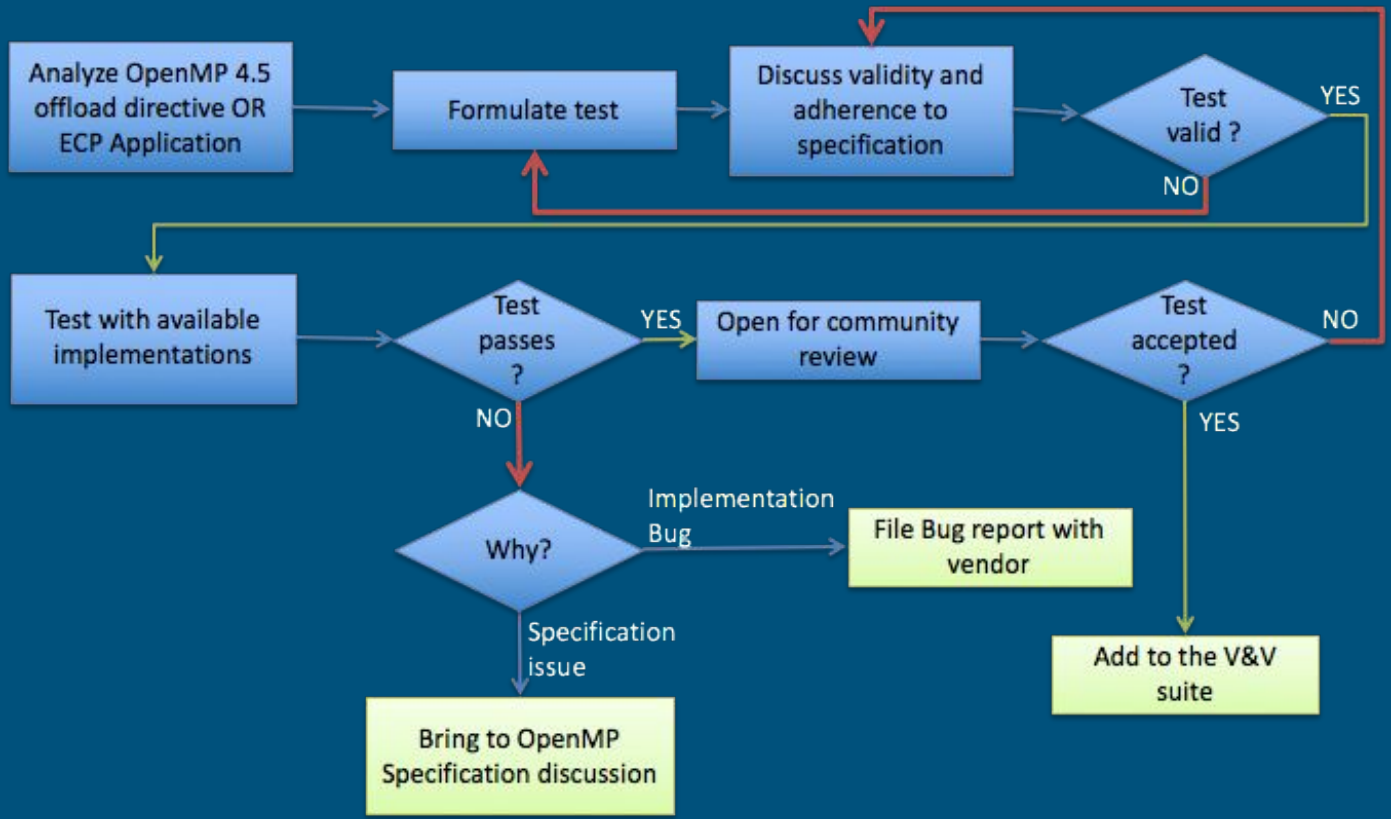
OpenMP Offloading programming model



Methodology

Validation of OpenMP 4.5 offloading features

Workflow



Validation of OpenMP 4.5 offloading features

Example C Code

```
1  int num_dev = omp_get_num_devices();
2  int h_matrix [N*num_dev];
3
4  for (int dev = 0; dev < num_dev; ++dev) {
5  #pragma omp target data map(from: h_matrix[dev*N:N]) device(dev)
6    {
7  #pragma omp target map(from: h_matrix[dev*N:N]) device(dev)
8      {
9        for (int i = 0; i < N; ++i)
10         h_matrix[dev*N + i] = dev;
11     } // end target
12   } // end target data
13 } // end for loop
14
15 // checking results
16 for (int dev = 0; dev < num_dev; ++dev) {
17   for (int i = dev*N ; i < (dev+1)*N; ++i)
18     OMPVV_TEST(errors, dev != h_matrix[i]));
19 }
```

Get number of available devices

Map data from selected device to host

Create target region on selected device

Computation offloaded to selected device

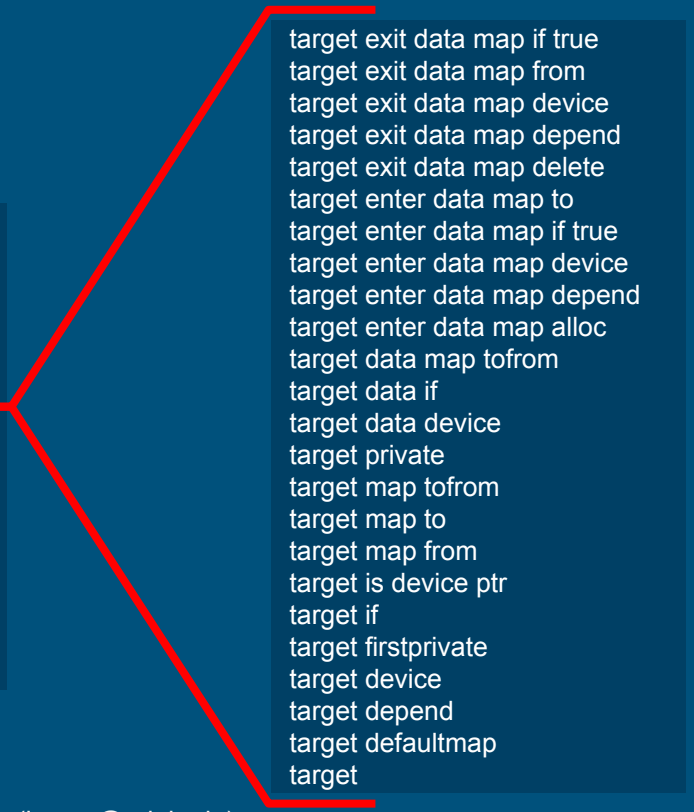
Checking data mapping and code offloading on multiple devices

Compare results with expected

Comparing runtime overhead

pseudocode

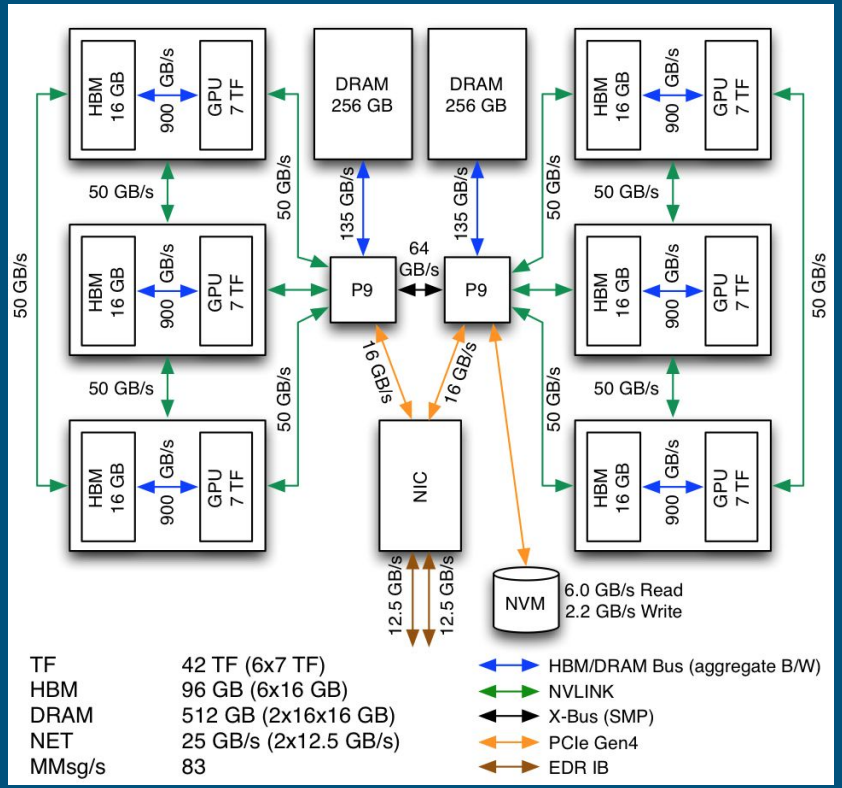
```
OMPVV_INIT_TEST;
for ( i = 0 ; i < NUM_REP ; i ++) {
  OMPVV_START_TIMER;
#pragma omp . . .
    OMPVV_TEST_LOAD; // if necessary
  OMPVV_STOP_TIMER;
  OMPVV_REGISTER_TEST;
}
OMPVV_PRINT_RESULT;
```



Experimental setup

System	Model	Processors	Cores/node	Threads/node	Memory	Accelerator	Complers
Titan	Cray XK7	AMD Opteron 6274	16	16	32 GB	1 NVIDIA K20X	CCE 8.7.2
Summitdev	IBM S822LC	2x Power8	20	160	256 GB	4 NVIDIA P100	GCC 7.1.1 Clang 3.8.0 XLC 13.1.6
Summit	IBM AC922	2x Power9	42	168	512 GB	6 NVIDIA V100	Clang 3.8.0 XLC 13.1.7
In-House	Generic	2x Intel Xeon E5-2670	16	32	64 GB	1 NVIDIA K20X	Clang 7.0.0 GCC 8.1.0

Experimental setup
Summit's Node



Results

*new results since paper publication

Summary of compiler supported features

	OpenMP Feature	Summitdev GCC 7.1.1	Summitdev gfortran 7.1.1*	Summitdev Clang 3.8.0	Summitdev XLC 13.1.6.	Summitdev XLF* 15.1.7	Summit Clang 3.8.0	Summit XLC 13.1.7	Titan CCE 8.7.2
	target	14/14	13/13	14/14	13/14	12/13	12/14	11/14	13/14
	target data	5/6	4/4	6/6	6/6	2/4	6/6	6/6	3/6
	target enter/exit data	6/7	-	6/7	6/7	-	6/7	6/7	5/7
	target enter data	6/7	-	6/7	6/7	-	6/7	6/7	5/7
	target update	5/5	-	5/5	4/5	-	5/5	4/5	4/5
*	target teams distribute	10/11	-	8/11	10/11	-	-	-	9/11
*	target teams distribute parallel for	13/14	-	11/14	11/14	-	-	-	10/14

Example of problematic features
XL and defaultmap

Specifications (Section 2.15.5 - page 216):

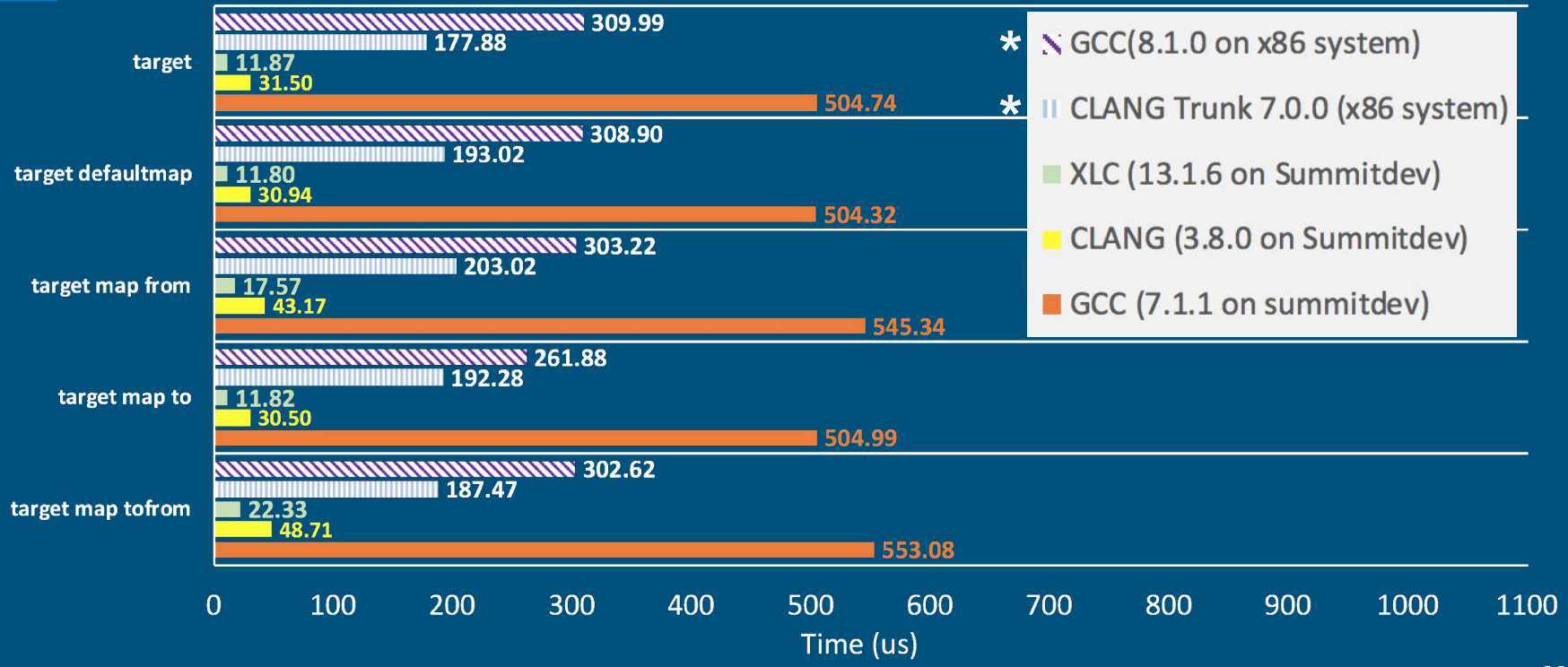
- If a **`defaultmap(tofrom:scalar)`** clause is not present then a scalar variable is not mapped, but instead has an implicit data-sharing attribute of firstprivate (see Section 2.15.1.1 on page 179).

Test:

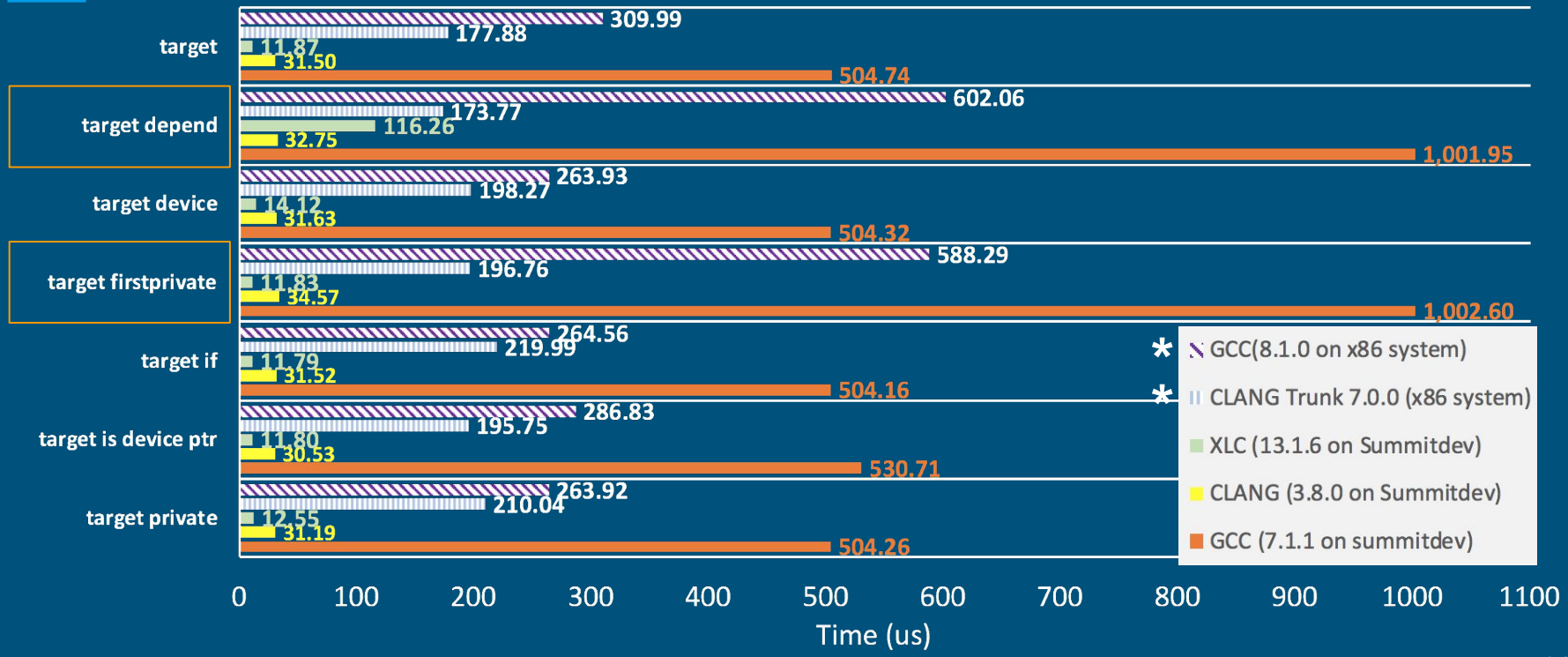
```
enum { VAL1 = 1, VAL2, VAL3, VAL4} scalar_enum = VAL1
#pragma omp target
  {
    scalar_enum = VAL4;
  }
OMPVV_TEST_AND_SET_VERBOSE(errors, scalar_enum != VAL1);
```

Failed condition

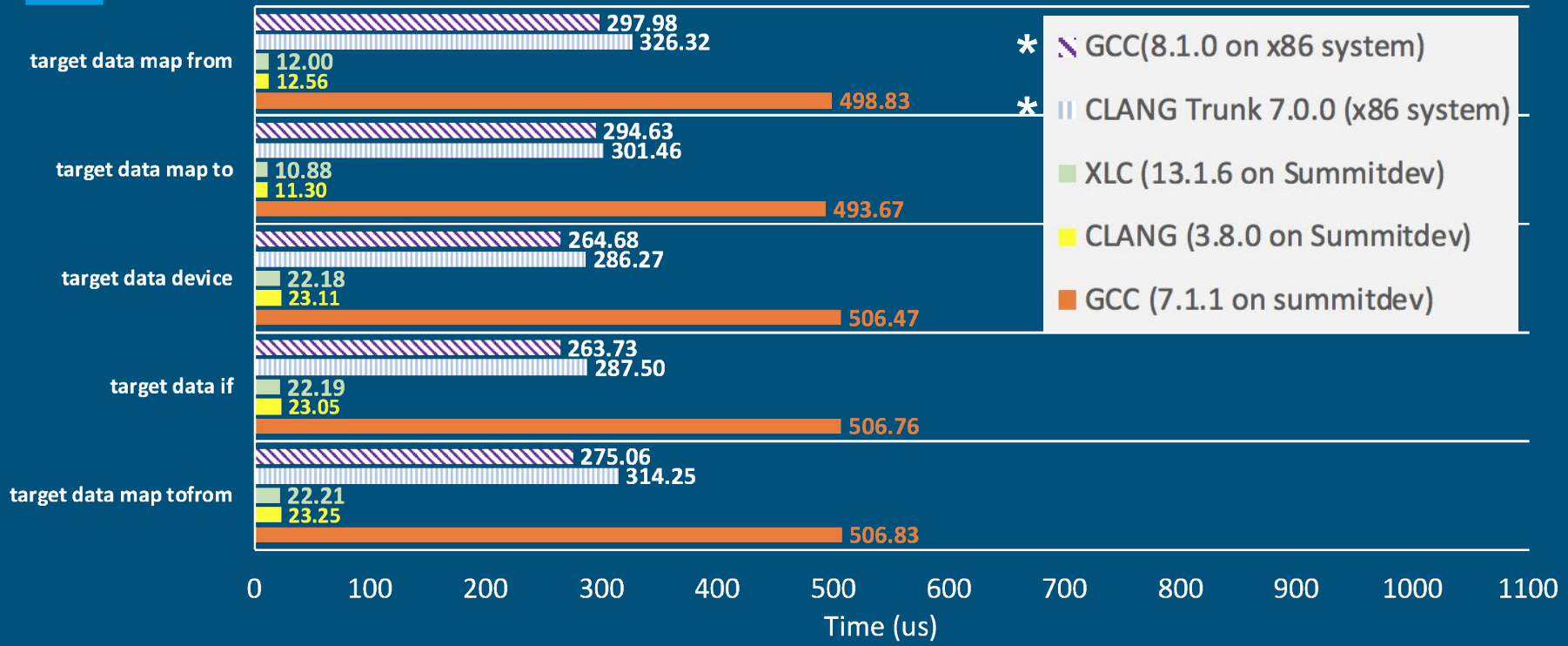
Target directive



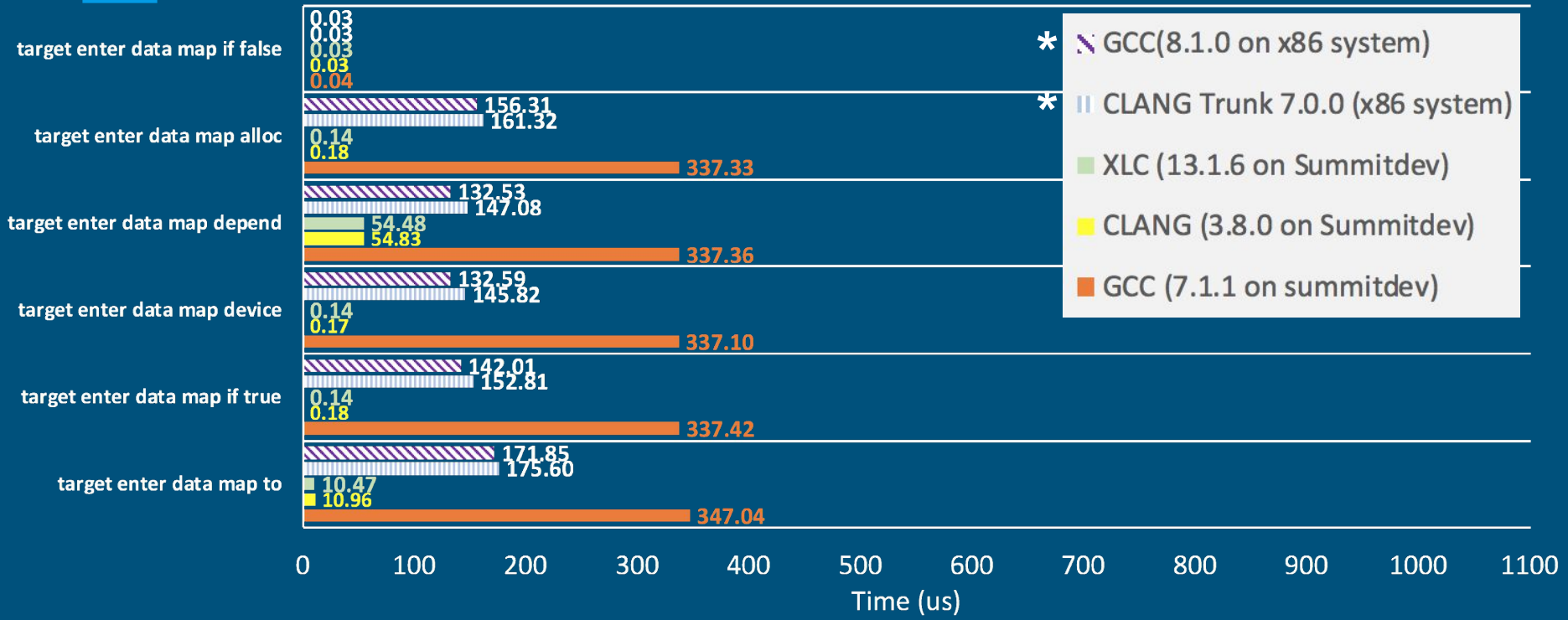
Target directive



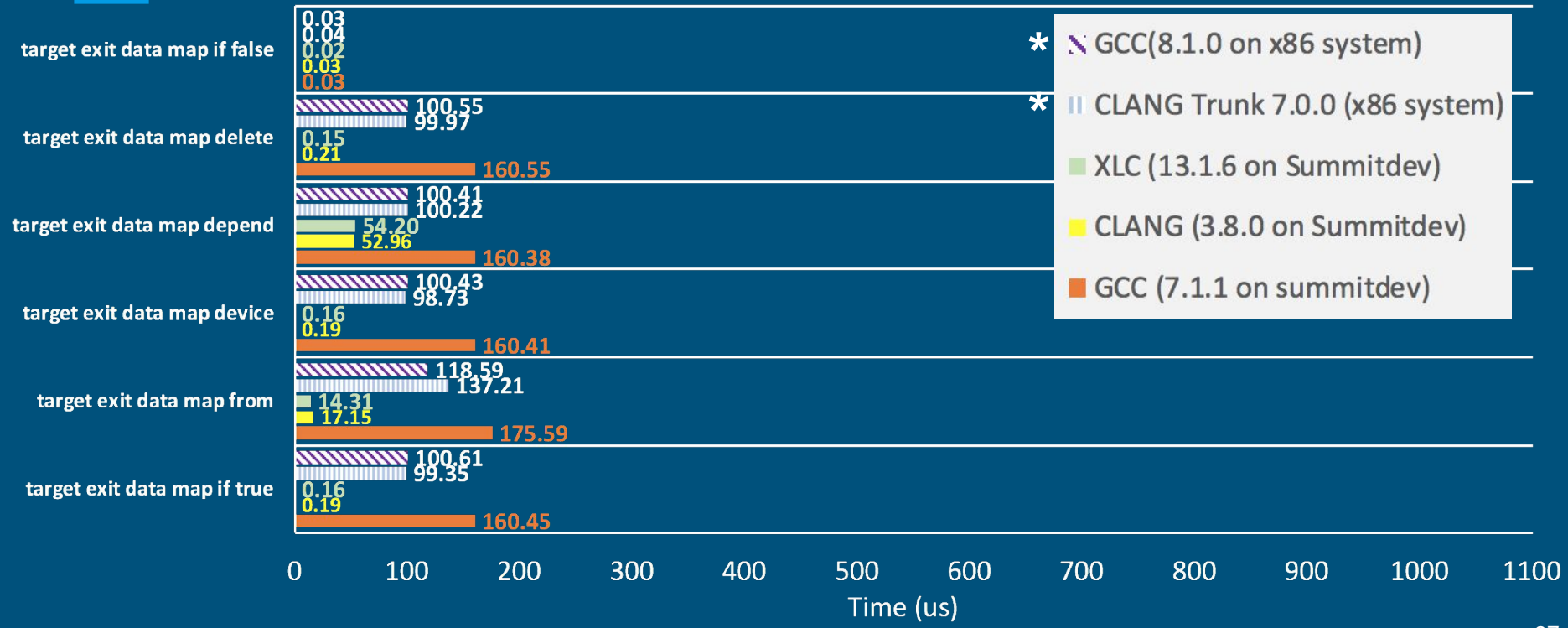
Target data directive



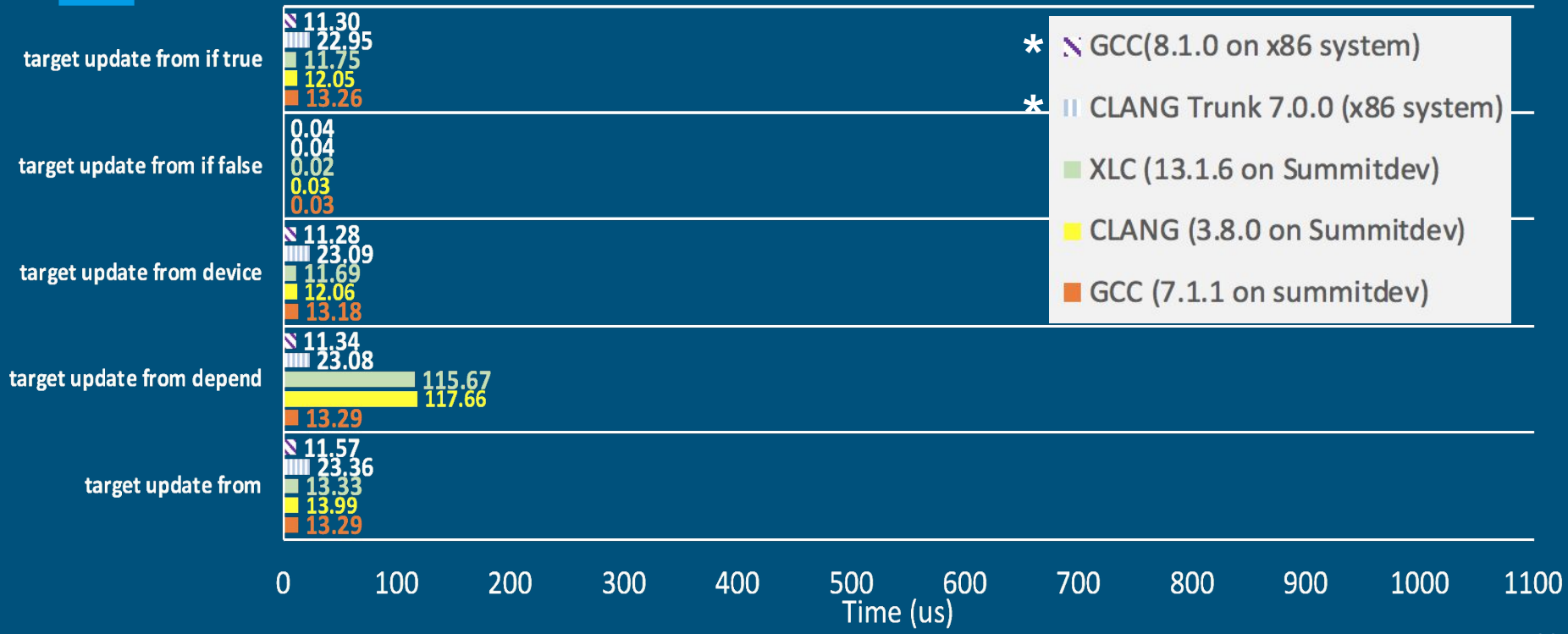
Target enter/exit data directive



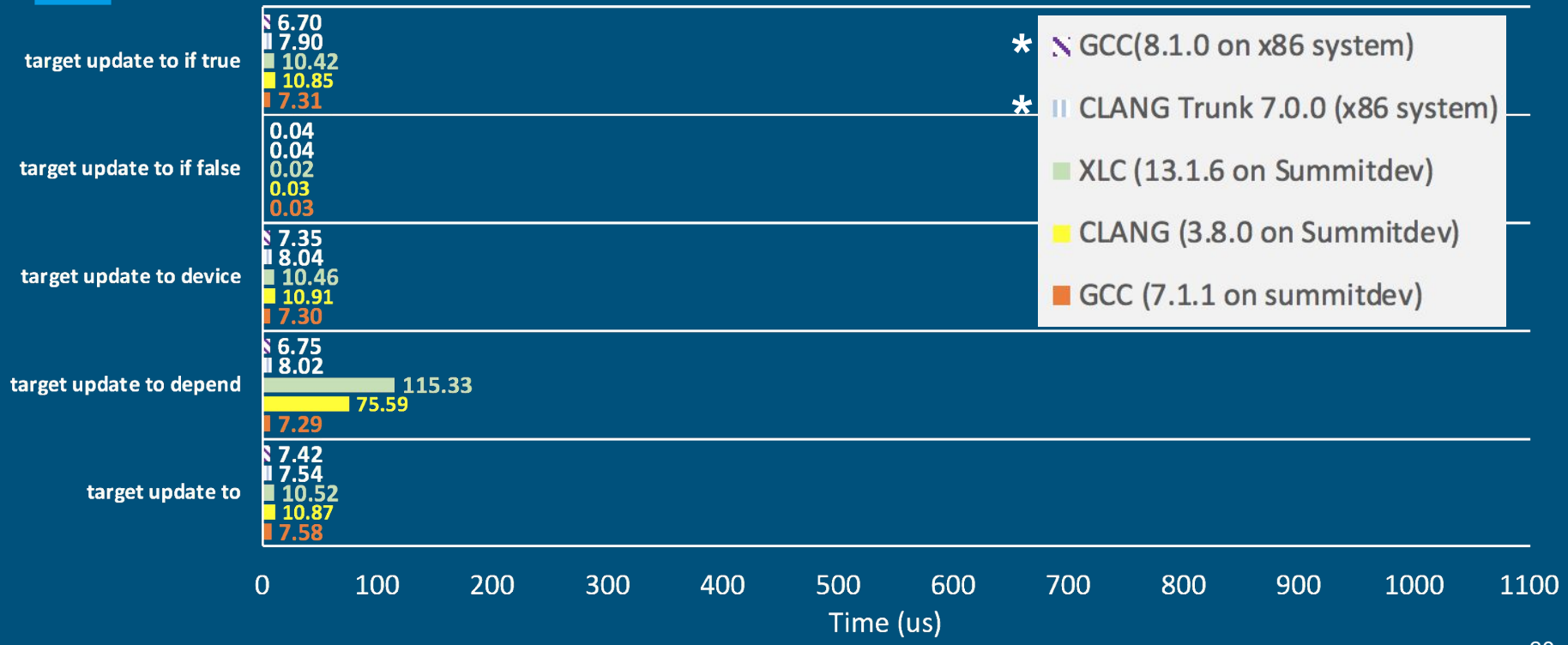
Target enter/exit data directive



Target update directive



Target update directive



Conclusions

Conclusions

- We have shown how use of accelerators and heterogeneous systems are the current trend in HPC and most likely will continue to increase.
- As a result programming models are fastly adapting to these new architectural features.
- We have shown the importance of a proper methodology to assess support and status of current compiler implementations (GCC, Clang/LLVM, IBM XL and Cray CCE) and the latest DOE Systems.
- We have shown a possible methodology, and the results of applying it to the OpenMP 4.5 specifications, emphasizing offloading features identified in DOE applications.
- As the development continues, we have seen compiler developers fastly adapting and responding to bug reports. We appreciate their effort and responsiveness.

Visit our website
https://crpl.cis.udel.edu/ompvvsollve/

Contact information:
Jose Monsalve (josem@udel.edu)
Swaroop Pophale (pophaless@ornl.gov)
Kyle Friedline (utimatu@udel.edu)
Oscar Hernandez (oscar@ornl.gov)
Sunita Chandrasekaran (schandra@udel.edu)

UNIVERSITY OF DELAWARE
ECP EXASCALE COMPUTING PROJECT
OAK RIDGE National Laboratory

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.**

Copyright

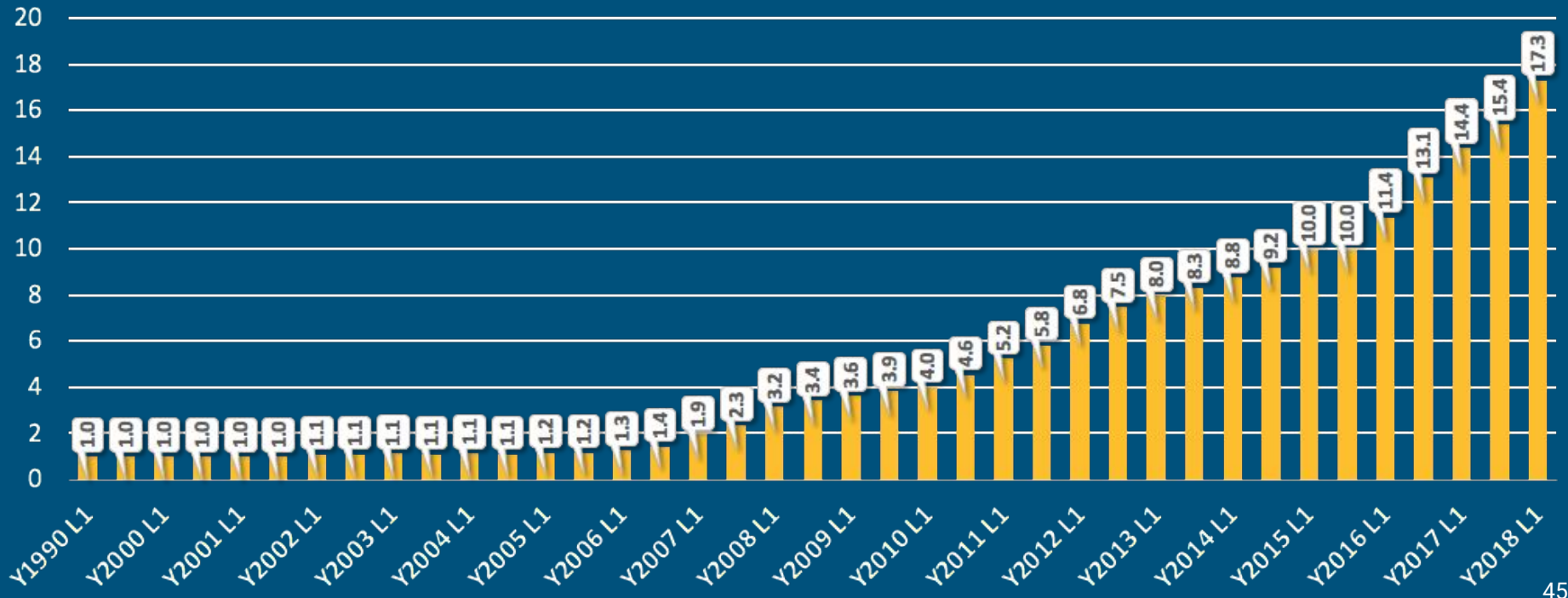
Gavel taken from(Chris Potter - Modifier: Ibrahim.ID): https://commons.wikimedia.org/wiki/File:3D_png_Judges_Gavel.png

Logos under CC0:

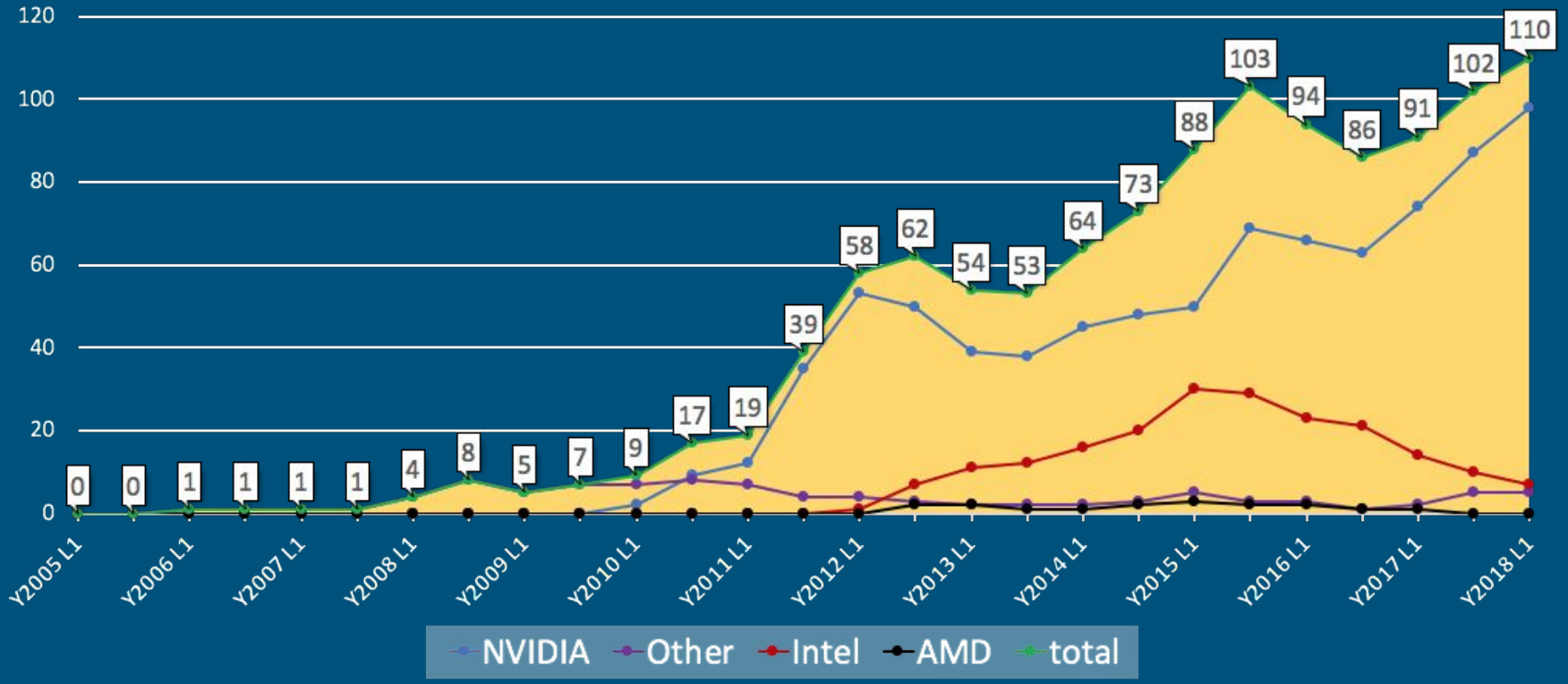
- https://pixabay.com/en/papers-stack-heap-documents-576385/
- https://pixabay.com/en/gear-wheel-gearwheel-gear-cogs-310906/
- https://pixabay.com/en/user-top-view-office-keyboard-154199/
- https://pixabay.com/en/brain-cognition-design-art-2029391/
- https://pixabay.com/en/computer-workstation-server-monitor-158743/
- https://pixabay.com/en/cpu-processor-intel-amd-chip-152656/
- https://pixabay.com/en/microprocessor-processor-cpu-chip-152599/
- https://pixabay.com/en/computer-cyber-circuitry-circuits-3163436/
- https://pixabay.com/en/ram-computer-memory-module-148579/
-

Back up slides

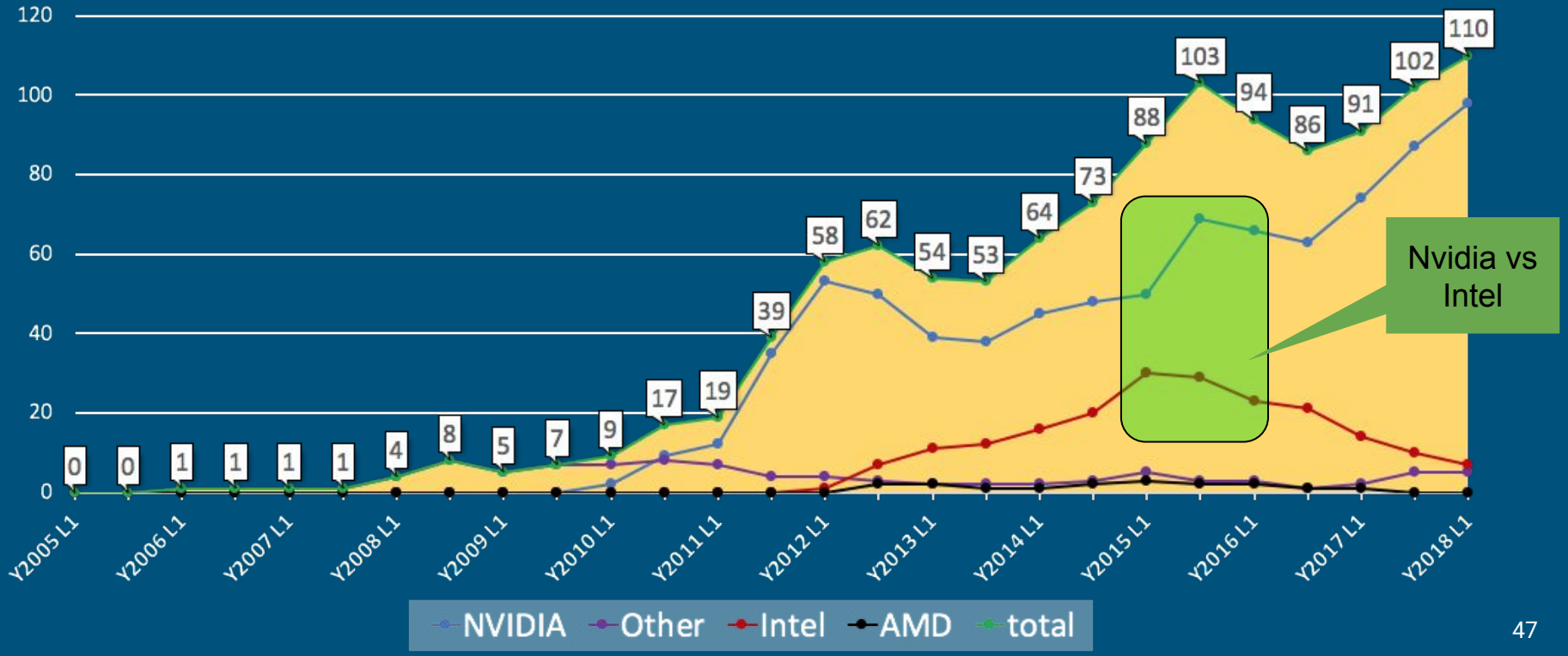
Top 500: Average number of cores per socket



Top 500: Number of systems with accelerators



Top 500: Number of systems with accelerators



Validation of OpenMP 4.5 offloading features

Design and write tests for OpenMP 4.5 Offloading features:

1. Study the specifications
2. Formulate testing methodology
3. Write initial test implementation
4. Discuss tests with team
5. Make corrections, report bugs or request clarifications
6. Run and report test result on multiple compilers available to us
    a. Main focus on DOE Systems