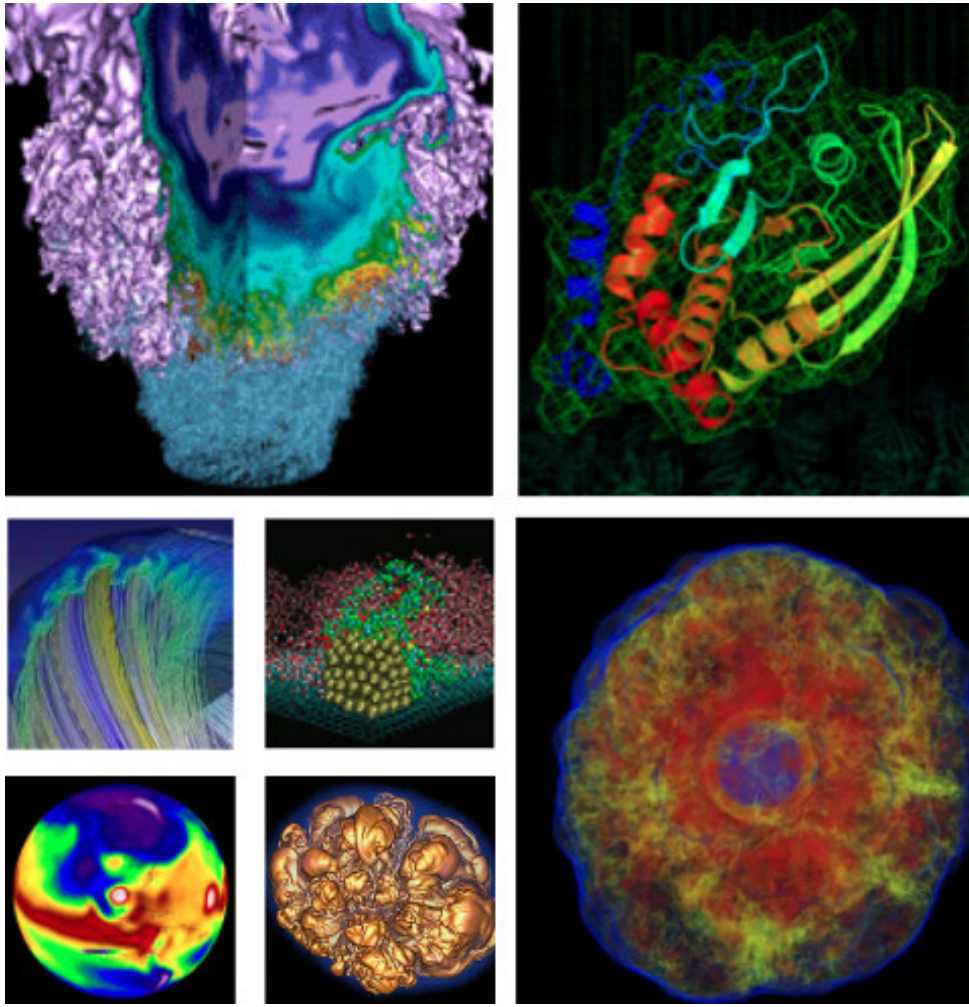
Unifying OpenMP offload support



National Energy Research Scientific Computing Center

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Rahulkumar Gayatri

National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory

April 1, 2019

✉ rgayatri@lbl.gov

OpenMP to target GPUs

- 5 of top 10 supercomputers are GPU based machines
  - Most of the codes optimized for CPUs have to now be rewritten to take advantage of the graphics card
- OpenMP 4.5 supports GPU offloading
  - Port incrementally for big codes
  - Control the parallelization via compile time flags
  - Need not be concerned with optimizations for newer architectures
- Bottleneck - Find compilers that support OpenMP 4.5

Why Attend this Talk

1. OpenMP 4.5 directives
2. This talk would provide a detailed analysis of the current state of OpenMP 4.5 implementations
   - Supported compilers
   - Differences in compiler implementations
3. Performance portability
   - Interpretations of OpenMP 4.5 directives on CPUs

OpenMP offloading to GPU

- OpenMP 4.5 compilers
  - XL/16.1(IBM)
  - Clang - LLVM-9.0
  - GCC/8.1
  - Cray/6.0

Volta GPU available on Cori and Summit

Target architecture - Volta

OpenMP 4.5 directives

OpenMP directives to offload code-blocks onto GPUs

Directives to distribute work across GPU threads

```
#pragma omp target //offload code block onto GPU-accelerator
{
    #pragma omp teams distribute //Distribute across threadblocks
    for()
    {
        #pragma omp parallel for simd//Distribute across threads
        for()
        {
        }
    }
}
```

Directives to move data to/from device/host

Clauses to use with **target** directives

```
map(to:...)    map(from:...)    map(tofrom:...)
```

Allocate and delete data on the device

```
#pragma omp target enter data map(alloc: list-of-data-structures[:])
#pragma omp target exit data map(delete: list-of-data-structures[:])
```

Update data on device and host

```
#pragma omp target update to/from (list-of-data-structures[:])
to - HostToDevice
from - DeviceToHost
```

OpenMP 4.5 directives to offload routines on the device

Routines

```
#pragma omp declare target
void foo(); //create a __device__ version of the routine
#pragma omp end declare target
```

Not necessary if routines are inlined

Differences in Compiler Implementations

OpenMP 4.5 directives map onto hardware

	Grid	**Thread**
GCC	teams distribute	parallel for simd
XL	teams distribute	parallel for
Clang	teams distribute	parallel for
Cray	teams distribute	simd

Table 1: **OpenMP 4.5 mapping onto GPU hardware**

c/c++ issues with offloading

- **XL** - older versions do not offload class operators
  - Make a copy of the routines to do the required operation
- **Cray** - Does not support printf inside target routines

Cheat Sheet of Do's and Dont's

- **XL**
  - Everything accessed inside the **target** region has to be mapped explicitly via **map** clauses
    - Even if they are allocated on the device beforehand
- **Clang**
  - Do not pass the same data to two different clauses in the same directive
  - Even if one of them is a **reduction** clause
- **GCC, Cray**
  - Always pass the directionality information to the **reduction** variables via **map** clauses
- **Clang** - Do not use **simd**

Interpretations of OpenMP 4.5 Directives on CPUs

Interpretation of OpenMP 4.5 directives on CPU

- XL/16.1
- **teams** - create as many teams as the number of threads
- Ignores other OpenMP 4.5 related directives, for example device memory allocation directives

XL-offload on P0

OpenMP 3.0 Threads = 128
OpenMP 4.5 Teams = 128
OpenMP 4.5 Threads = 1

Interpretation of OpenMP 4.5 directives on CPU

- Clang/LLVM/9.0, GCC/8.1, Intel/2018
- **teams** - create as many teams as the number of threads
- Ignores other OpenMP 4.5 related directives, for example device memory allocation directives

Offload on KNL

OpenMP 3.0 Threads = 272
OpenMP 4.5 Teams = 1
OpenMP 4.5 Threads = 272

Conclusions of the work

- No uniformity in OpenMP 4.5 directive mapping on GPU hardware among compilers
- No uniformity in the interpretation of compiler directives on CPUs
- Moving target and hence a lot of small issues are not fixed