OpenMP and NVIDIA

- OpenMP is the dominant standard for directive-based parallel programming.
- NVIDIA joined OpenMP in 2011 to contribute to discussions around parallel accelerators.
- NVIDIA proposed the TEAMS construct for accelerators in 2012.
- OpenMP 4.0 with accelerator support released in 2013.
Why does NVIDIA care about OpenMP?
3 Ways to Accelerate Applications

Applications

Libraries
“Drop-in” Acceleration

Compiler Directives
Easily Accelerate Applications

Programming Languages
Maximum Flexibility
Reaching a Broader Set of Developers

1,000,000’s

100,000’s

Early Adopters

Time

Present

Universities
Supercomputing Centers
Oil & Gas
Research

CAE
CFD
Finance
Rendering
Data Analytics
Life Sciences
Defense
Weather
Climate
Plasma Physics

2004
OpenMP PARALLEL FOR

Executes the iterations of the next for loop in parallel across a team of threads.

```c
#pragma omp parallel for
for (i=0; i<N; i++)
    p[i] = v1[i] * v2[i];
```
OpenMP TARGET PARALLEL FOR

- **Offloads** data and execution to a target device, then
- Executes the iterations of the next for loop *in parallel* across a *team* of *threads*.

```c
#pragma omp target
#pragma omp parallel for
for (i=0; i<N; i++)
    p[i] = v1[i] * v2[i];
```
OpenMP 4.0 TEAMS/DISTRIBUTE

- Creates a league of teams on the target device, distributes blocks of work among those teams, and executes the remaining work in parallel within each team.

- This code is portable whether 1 team or many teams are used.

```c
#pragma omp target teams
#pragma omp 
    distribute parallel for 
    reduction(+:sum)
for (i=0; i<N; i++)
    sum += B[i] * C[i];
```
#pragma omp target

#pragma omp parallel for reduction(+:sum)
for (i=0; i< N; i++) sum += B[i] * C[i];
OpenMP 4.0 Teams Distribute Parallel For

```c
#pragma omp target teams
#pragma omp distribute parallel for
#pragma omp parallel for reduction(+:sum)
for (i=0; i< ?; i++) sum += B[i] * C[i];
#pragma omp parallel for reduction(+:sum)
for (i=?; i< ?; i++) sum += B[i] * C[i];
#pragma omp parallel for reduction(+:sum)
for (i=?; i< ?; i++) sum += B[i] * C[i];
#pragma omp parallel for reduction(+:sum)
for (i=?; i< N; i++) sum += B[i] * C[i];
```

The programmer doesn’t have to think about how the loop is decomposed!
OMP + NV: We’re not done yet!
OMP + NV: We’re not done yet!

- Hardware parallelism is not going away, programmers demand a simple, portable way to use it.
- OpenMP 4.0 is just the first step toward a portable standard for directive-based acceleration.
- We will continue to work with OpenMP to address the challenges of parallel computing:
  - Improved Interoperability
  - Improved Portability
  - Improved Expressibility
Thank You