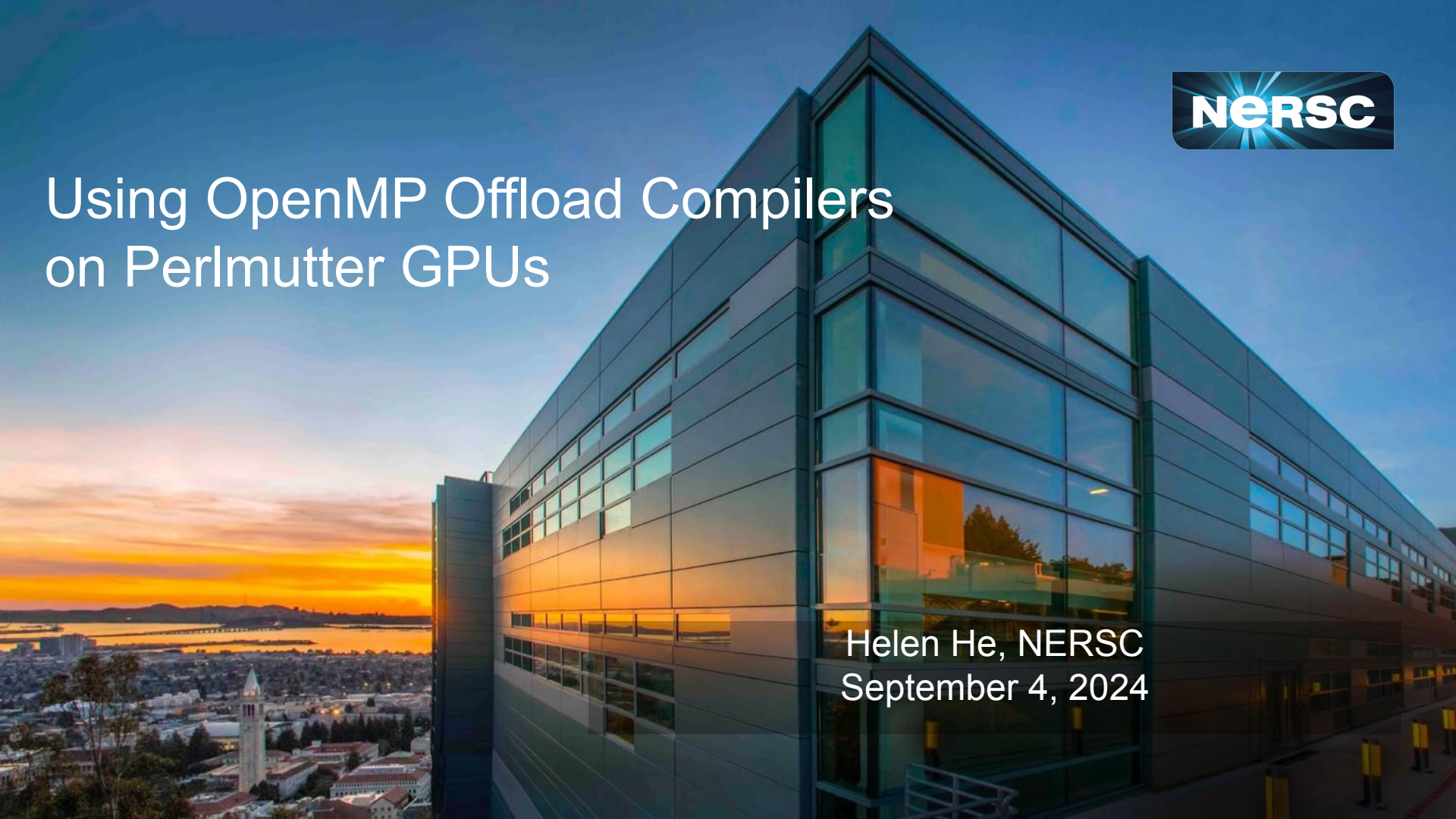

Using OpenMP Offload Compilers on Perlmutter GPUs

Helen He, NERSC
September 4, 2024

Perlmutter OpenMP Offload Compiler Support

- Vendor provided and supported
  - NVHPC
  - CCE
- Community (Open Source)
  - LLVM/Clang
  - GCC

Using NVHPC Compiler on Perlmutter GPU

% module load PrgEnv-nvidia

Supports C/C++/Fortran. Use compiler wrappers
-Minfo is optional which provides compile time info
% cc -fast -mp=gpu -Minfo=mp,accel src.c
% CC -fast -mp=gpu -Minfo=mp,accel src.cc
% ftn -fast -mp=gpu -Minfo=mp,accel src.f90

Optional runtime messages
% export NVCOMPILER_ACC_NOTIFY=<value>
where value is 1: kernel launches 2: data transfers
4: region entry/exit
8: wait operations or synchronizations with the device
16: device memory allocates and deallocates

Using CCE Compiler on Perlmutter GPU

% module load PrgEnv-cray

Supports C/C++. Use compiler wrappers (cc/CC/ftn)
% cc -Ofast -fopenmp src.c
% CC -Ofast -fopenmp src.cc
% ftn -O3 -h omp -h noacc src.f90

Optional runtime messages
% export CRAY_ACC_DEBUG=<value> where value can be 1, 2, 3

Note
- ➢ CCE C++ is based on LLVM/Clang; CCE Fortran is classic Cray Compiler
- ➢ simd clause needed in CCE Fortran for thread parallelism on GPU

Using LLVM/Clang Compiler on Perlmutter GPU

% module load PrgEnv-llvm

Supports C/C++. Use native compilers

% clang -Ofast -fopenmp -fopenmp-targets=nvptx64 src.c

% clang++ -Ofast -fopenmp -fopenmp-targets=nvptx64 src.cc

Optional runtime messages

% export LIBOMPTARGET_INFO=-1

Using GCC Compiler on Perlmutter GPU

```
% module load PrgEnv-gnu
% module use /global/cfs/cdirs/nstaff/rgayatri/modules
% module load gcc-offload/13.2.0
```

Supports C/C++/Fortran. Use native compilers

```
% gcc -Ofast -fopenmp -foffload=nvptx-none="-Ofast -lm -latomic -misa=sm_80"  src.c
% g++ -Ofast -fopenmp -foffload=nvptx-none="-Ofast -lm -latomic -misa=sm_80" src.cc
% gfortran -Ofast -fopenmp -foffload=nvptx-none="-Ofast -lm -latomic -misa=sm_80" src.f90
```

Optional runtime messages

```
% export GOMP_DEBUG=1
```

Warning

- ➢ OpenMP target offload performance not optimal; expect future improvement
- ➢ NERSC has done minimal testing only, can not offer good support

Using Perlmutter GPU

Perlmutter documentation: https://docs.nersc.gov/systems/perlmutter/

```
% ssh user-name@perlmutter.nersc.gov

# Get on a GPU node via salloc:
% salloc -C gpu -N 1 -q shared -c 32 -G 1 -t 1:00:00 -A ntrain5
<get on a GPU node>

% cc -fast -mp=gpu -Minfo=mp,accel mycode.c
% export OMP_NUM_THREADS=8    # for CPU OpenMP
./mycode.exe <args>
```

Sample Batch Script (non-MPI)

```
% cat myjob.sl
#!/bin/bash
#SBATCH -N 1
#SBATCH -C gpu
#SBATCH -t 10:00
#SBATCH -q shared
#SBATCH -c 32
#SBATCH -G 1
#SBATCH -A ntrain5
module load PrgEnv-nvidia
cc -fast -mp=gpu -Minfo=mp,accel -o mycode.exe mycode.c
export OMP_NUM_THREADS=8    # for CPU OpenMP
./mycode.exe <args>
```

```
% module load PrgEnv-nvidia
% sbatch myjob.sl
```

-A is required for GPU jobs
-q shared -c 32 -G 1 is to share the node

Perlmutter OpenMP Offload Compilers (as of 9/4/2024)

Compiler	Compile Command	Compile Flags	omp loop	Comment
NVHPC 23.9 C/C++/Fortran	cc/CC/ftn	-fast -mp=gpu	Yes	overall best supported and best performance. **Recommend for C/C++ and Fortran codes**
CCE 17.0.0 Fortran	ftn	-O3 -h omp -h noacc	Yes	classic Cray Compiler. **Recommend for Fortran codes**
CCE 17.0.0 C/C++	cc/CC	-Ofast -fopenmp	No	based on LLVM/Clang compiler. **Recommend for C/C++ codes**
LLVM 18.1.0 C/C++	clang/clang++	-Ofast -fopenmp -fopenmp-targets=nvptx64	No	**Recommend for C/C++ codes**
GCC 13.2.0 C/C++/Fortran	gcc/g++/gfortran	-Ofast -fopenmp -foffload=nvptx-none="-Ofast -lm -latomic -misa=sm_80"	Yes	performance not optimal as of now