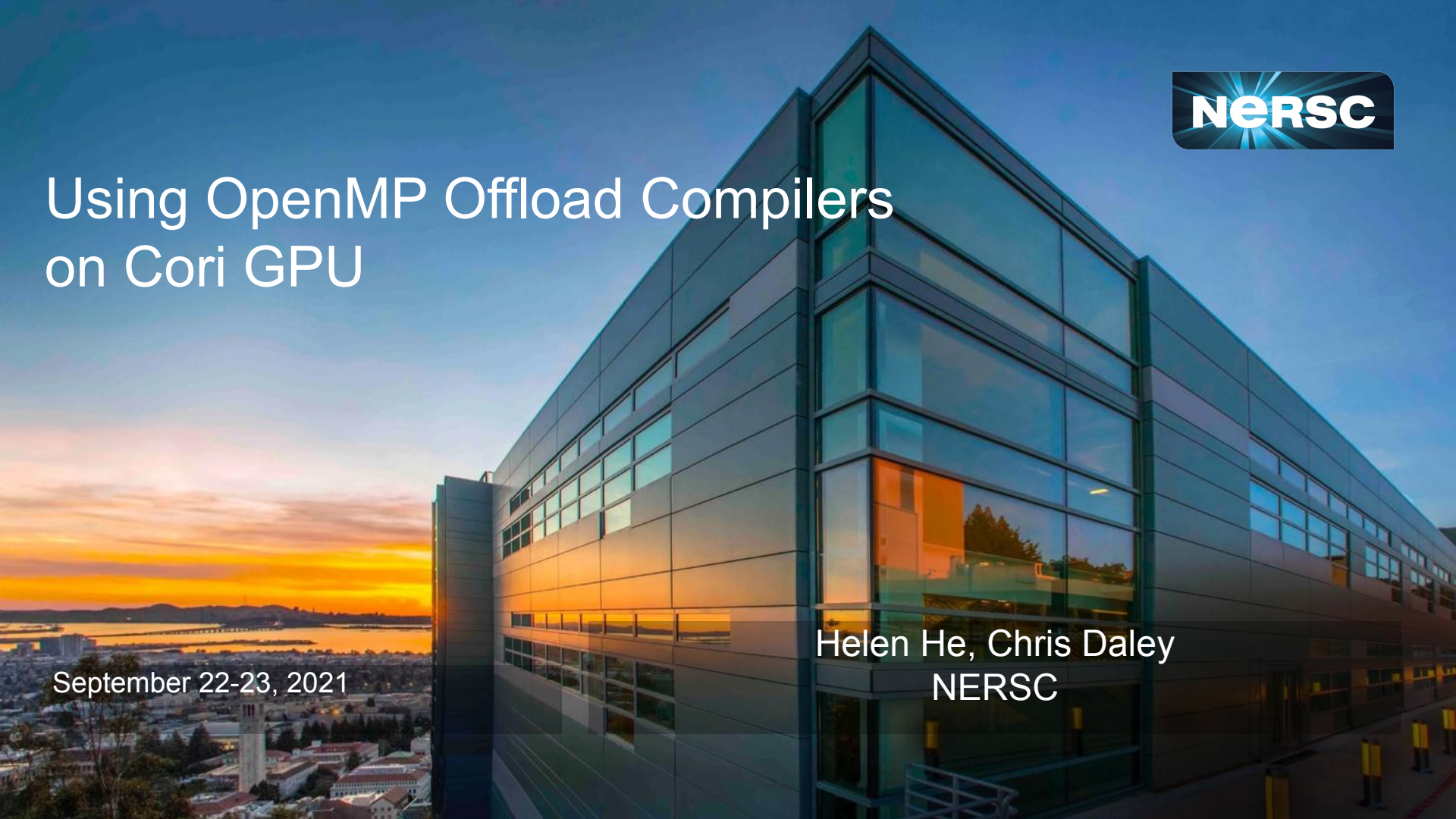

Using OpenMP Offload Compilers on Cori GPU

September 22-23, 2021

Helen He, Chris Daley
NERSC

Using Cori GPU

Cori GPU documentation: https://docs-dev.nersc.gov/cgpu/
Login to Cori
% ssh user-name@cori.nersc.gov

All build and run should be done on a GPU compute node:
% module purge && module load cgpu
# Get on a GPU node via salloc:
# During reservation hours (11am - 2pm Pacific on each day)
% salloc -C gpu -N 1 -c 10 -G 1 -t 1:00:00 -A ntrain --reservation=omp_offload_day1 -q shared
# Outside of reservation hours
% salloc -C gpu -N 1 -c 10 -G 1 -t 1:00:00
% cd C/4-openmp-gpu-datal
% module load nvhpc/21.7
% make
# Execute your application (non-MPI):
% export OMP_NUM_THREADS=8 (for CPU, use a value smaller than -c in salloc above)
% . /jacobi.C.nvhpc.exe <args>

Using nvhpc Compiler on Cori GPU

```
% module purge
% module load cgpu
% module load nvhpc/21.7
```

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

```
# -Minfo provides compile time info
% nvc -fast -mp=gpu -Minfo=mp,accel -gpu=cc70 src.c
% nvc++ -fast -mp=gpu -Minfo=mp,accel -gpu=cc70 src.cc
% nvfortran -fast -mp=gpu -Minfo=mp,accel -gpu=cc70 src.f90
```

Optional runtime messages

% export NVCOMPILER_ACC_NOTIFY=<value>

where value is 1: high level overview of kernels executed and data transferred
2: break down data transfer by each variable
3: both above

Using LLVM/Clang Compiler on Cori GPU

```
% module purge
% module load cgpu
% module load PrgEnv-llvm/13_rc3
```

Supports C/C++. Use native compilers

```
% clang -Ofast -fopenmp -fopenmp-targets=nvptx64-nvidia-cuda src.c
% clang++ -Ofast -fopenmp -fopenmp-targets=nvptx64-nvidia-cuda  src.cc
```

Optional runtime messages

```
% export LIBOMPTARGET_INFO=-1
```

Using CCE Compiler on Cori GPU

% module purge; module load cgpu
% module load PrgEnv-cray
% module load cdt
% module load craype-x86-skylake
% module switch cce cce/10.0.3 (can use cce/11.0.1 for Fortran)
% module load cuda/10.1.243
% module unload cray-libsci

or % source ../../Makefiles/Cori_setup_cce

Supports C/C++. Use compiler wrappers (cc/CC/ftn)
% cc -Ofast -fopenmp -fopenmp-targets=nvptx64 -Xopenmp-target=nvptx64 -march=sm_70 src.c
% CC -Ofast -fopenmp -fopenmp-targets=nvptx64 -Xopenmp-target=nvptx64 -march=sm_70 src.cc
% ftn -O3 -h omp -h noacc -haccel=nvidia70 src.f90

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

Using GCC Compiler on Cori GPU

```
% module purge
% module load cgpu
% module load gcc/11.2.0
```

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

```
% gcc -Ofast -fopenmp -foffload=nvptx-none="-Ofast -lm -latomic -misa=sm_35"  src.c
% g++ -Ofast -fopenmp -foffload=nvptx-none="-Ofast -lm -latomic -misa=sm_35" src.cc
% gfortran -Ofast -fopenmp -foffload=nvptx-none="-Ofast -lm -latomic -misa=sm_35" src.f90
```

Sample Batch Script (non-MPI)

```
% cat myjob.sl
#!/bin/bash
#SBATCH -C gpu
#SBATCH -t 1:00:00
#SBATCH -c 10
#SBATCH -G 1
#SBATCH -q shared
#SBATCH -A ntrain
#SBATCH --reservation=omp_offload_day1
module load nvhpc/21.7
cd C/4-openmp-gpu-data
nvc -fast -mp=gpu -Minfo=mp,accel -gpu=cc70 -o
jacobi.C.nvhpc.exe jacobi.c     # or just do: make
export OMP_NUM_THREADS=8    # for CPU OpenMP
./jacobi.C.nvhpc.exe <args>
```

```
% module purge; module load cgpu
% sbatch myjob.sl
```

-q, -A, --reservations flags are required when using the node reservations