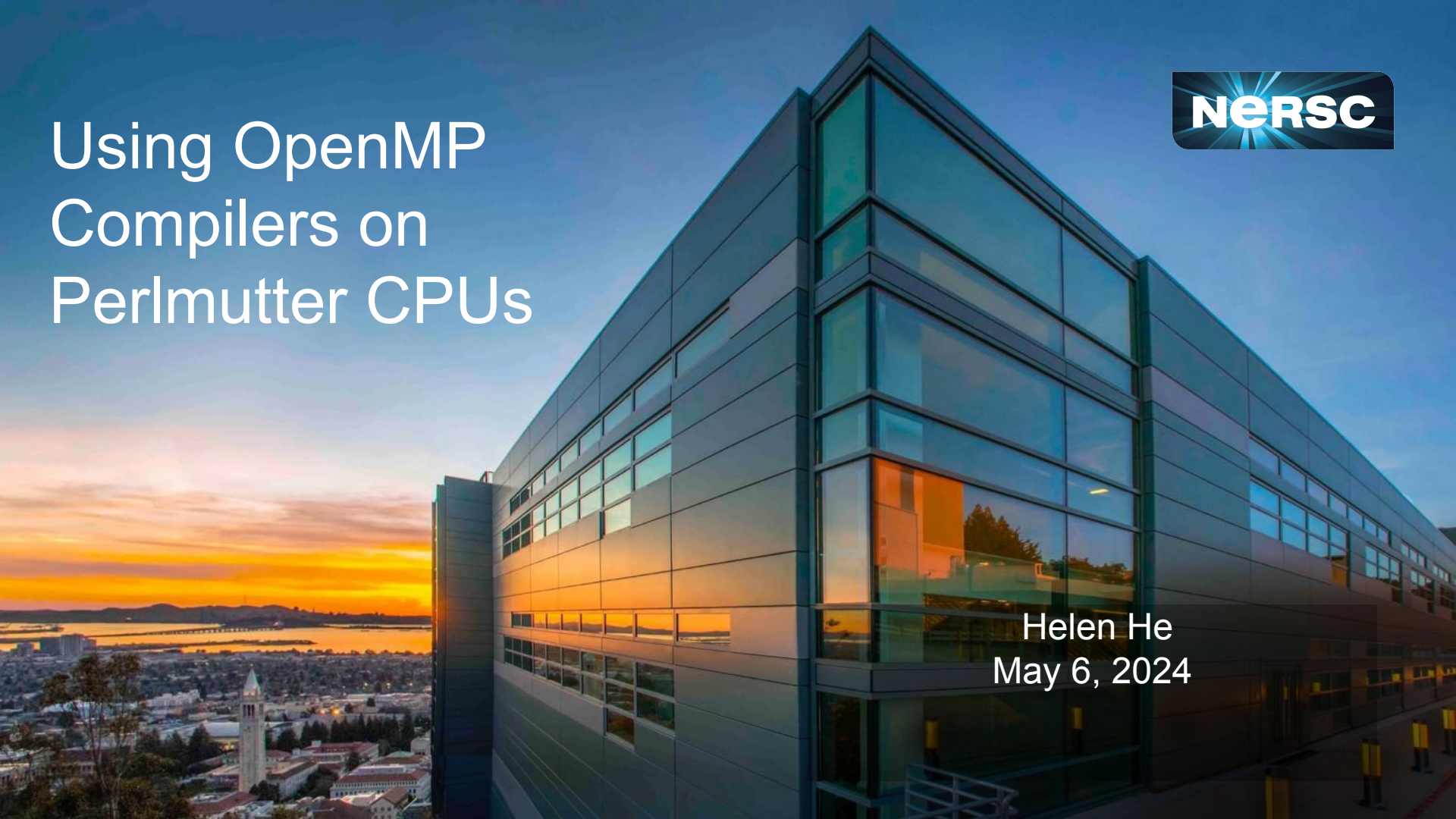
# Using OpenMP Compilers on Perlmutter CPUs



Helen He
May 6, 2024

# Using Perlmutter


## System Specifications

| Partition | # of nodes | CPU | GPU |
|---|---|---|---|
| GPU | 1536 | 1x AMD EPYC 7763 | 4x NVIDIA A100 (40GB) |
| | 256 | 1x AMD EPYC 7763 | 4x NVIDIA A100 (80GB) |
| CPU | 3072 | 2x AMD EPYC 7763 | - |
| Login | 40 | 1x AMD EPYC 7713 | 1x NVIDIA A100 (40GB) |

# Access to Perlmutter

- NERSC users can use your existing account
- Non-users were sent instructions to get a training account
  - Project: ntrain8
  - Training accounts for Session 1 valid through May 15
- Login to Perlmutter: ssh username@perlmutter.nersc.gov
- Getting homework exercises

```
% cd $SCRATCH
% git clone https://github.com/NERSC/openmp-series-2024
% cd openmp-series-2024
% cd Session-1-Introduction/exercises/cpp/0x_xx       (or: cd …./fortran/…)
% make
% sbatch xx.slurm
```

# GCC is Default Compiler on Perlmutter

- PrgEnv-gnu is the default programming environment;  GCC is the default compiler

```
[yunhe@perlmutter:login21:~> module list

Currently Loaded Modules:
  1) craype-x86-milan                      9) craype/2.7.30
  2) libfabric/1.15.2.0                   10) gcc-native/12.3
  3) craype-network-ofi                   11) perftools-base/23.12.0
  4) xpmem/2.6.2-2.5_2.38__gd067c3f.shasta 12) cpe/23.12
  5) PrgEnv-gnu/8.5.0                     13) cudatoolkit/12.2
  6) cray-dsmml/0.2.2                     14) craype-accel-nvidia80
  7) cray-libsci/23.12.5                  15) gpu/1.0
  8) cray-mpich/8.1.28
```

- User compiler wrappers to build: cc for C codes, CC for C++, and ftn for Fortran codes
  - It uses native gcc compilers (gcc, g++, and gfortran) underneath
- To compile an OpenMP code
  - cc -fopenmp -O3 mycode.c -o mycode.exe
  - CC -fopenmp -O3 mycode.cc -o mycode.exe
  - ftn -fopenmp -O3 mycode.f90 -o mycode.exe

# Other Available Compilers on Perlmutter

- Besides GCC, there are Nvidia, CCE, and Intel compilers available on Perlmutter under PrgEnv-gnu, PrgEnv-nvidia, PrgEnv-cray, and PrgEnv-intel respectively
- To use a different compiler, load the PrgEnv-xx module, and still use compiler wrappers to build, for example:
  - % module load PrgEnv-nvidia

```
yunhe@perlmutter:login21:~> module list

Currently Loaded Modules:
  1) craype-x86-milan                      9) gpu/1.0
  2) libfabric/1.15.2.0                   10) nvidia/23.9          (g,c)
  3) craype-network-ofi                   11) craype/2.7.30        (c)
  4) xpmem/2.6.2-2.5_2.38__gd067c3f.shasta 12) cray-dsmml/0.2.2
  5) perftools-base/23.12.0               13) cray-mpich/8.1.28    (mpi)
  6) cpe/23.12                            14) cray-libsci/23.12.5  (math)
  7) cudatoolkit/12.2                     15) PrgEnv-nvidia/8.5.0  (cpe)
  8) craype-accel-nvidia80
```

  - % cc -fopenmp -O3 mycode.c  (or: cc -mp mycode.c )
  - % module load PrgEnv-intel
  - % ftn -fopenmp -O3 mycode.f90  (or: ftn -qopemp mycode.f90)

# Running Jobs on Perlmutter CPUs

- Slurm batch scheduler is used to schedule jobs
- You can run a batch queue job: prepare and submit a batch script

  % sbatch myjob.slurm

- You can run an interactive batch job

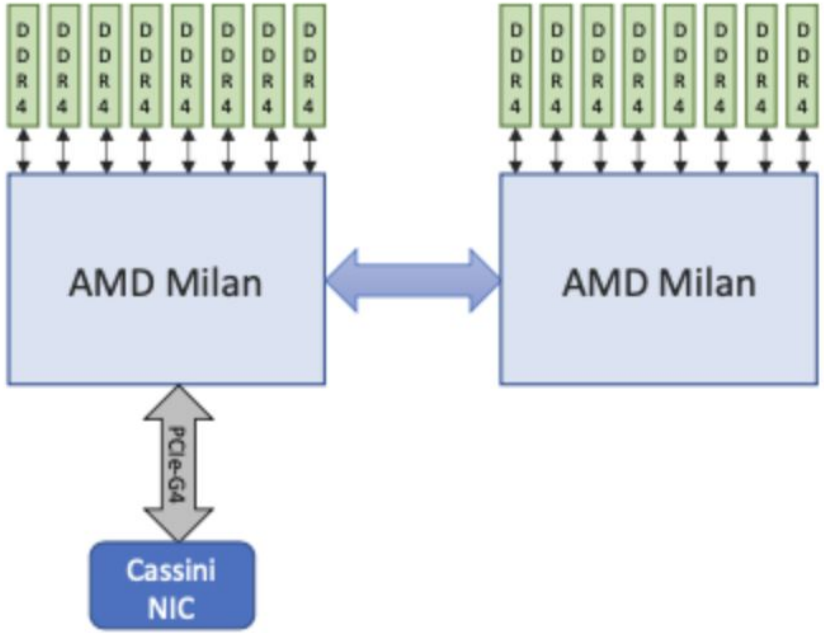
  % salloc -N 1 -q interactive -C cpu -t 30:00
  <will land on a compute node>
  % export OMP_NUM_THREADS=8
  % ./mycode.exe

**Sample pure OpenMP batch script**

```
#!/bin/bash
#SBATCH -q debug
#SBATCH -N 1
#SBATCH -t 10:00
#SBATCH -C cpu
#SBATCH -J mycode
#SBATCH -o mycode_%j.out

export OMP_NUM_THREADS=8
./mycode.exe
```

# Perlmutter CPU nodes




- Each CPU node has: 2x AMD EPYC 7763 (Milan) CPUs, and 64 cores per CPU, meaning: 128 physical cores per CPU
- With 2 hyperthreads per core, meaning: 256 logical cores total (Slurm sees this when scheduling jobs)
- Without explicitly setting OMP_NUM_THREADS, you will see 256 threads running with most compilers