



Using OpenMP Offload Compilers on Perlmutter

September 29, 2023

Helen He, Rahul Gayatri, Paul Lin
NERSC

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
```

```
or % source ../../Makefiles/pm_setup_cce
```

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
```

```
or % source ../../Makefiles/pm_setup_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/m1759/wwei/Modules/perlmutter/modulefiles  
% module load gcc/13.1.0
```

```
or % source ../../Makefiles/pm_setup_gcc
```

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 hands-on exercises:

% git clone <https://github.com/olcf/openmp-offload.git>

Get on a GPU node via salloc:

During reservation hours (10:30 am - 1pm Pacific on each day)

% salloc -C gpu -N 1 -q shared -c 32 -G 1 -t 1:00:00 -A ntrain4 --reservation=omp_sept29

Outside of reservation hours

% salloc -C gpu -N 1 -q shared -c 32 -G 1 -t 1:00:00 -A ntrain4

% cd openmp-offload/C/4-openmp-gpu-data

% module load PrgEnv-nvidia

(or: % source ../../Makefiles/pm_setup_nvhpc)

% 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>

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 ntrain4
#SBATCH --reservation=omp_day2

module load PrgEnv-nvidia
cd ../openmp-offload/C/4-openmp-gpu-data
cc -fast -mp=gpu -Minfo=mp,accel -o jacobi.C.nvhpc.exe
jacobi.c # or just do: make

export OMP_NUM_THREADS=8 # for CPU OpenMP

./jacobi.C.nvhpc.exe <args>
```

```
% cd C/4-openmp-gpu-data
% module load PrgEnv-nvidia
% make
% sbatch myjob.sl
```

-A, --reservation flags are required
when using the node reservations
-q shared -c 32 -G 1 is to share the
node



BERKELEY LAB
Bringing Science Solutions to the World



U.S. DEPARTMENT OF
ENERGY

Office of
Science

Perlmutter OpenMP Offload Compilers (as of Sept 2023)

Compiler	Compile Command	Compile Flags	NERSC Preference	omp loop	Comment
NVHPC 23.1 C/C++/Fortran	cc/CC/ftn	-fast -mp=gpu	1	Yes	overall best supported and best performance
CCE 15.0.1 Fortran	ftn	-O3 -h omp -h noacc	2	Yes	classic Cray Compiler
CCE 15.0.1 C/C++	cc/CC	-Ofast -fopenmp	3	No	based on LLVM/Clang compiler
LLVM 16.0.0 C/C++	clang/clang++	-Ofast -fopenmp -fopenmp-targets=nvptx64	4	No	
GCC 13.1.0 C/C++/Fortran	gcc/g++/gfortran	-Ofast -fopenmp -foffload=nvptx-none="-Ofast -lm -latomic -misa=sm_80"	5	Yes	performance not optimal as of now