Programming Environments & Compilation

Starting at 11:05 am PST
Outline: Four Main Ways of Getting Software

- Loading modules
- Containers
- Compiling from source
- Conda, Spack & E4S

Welcome to perlmutter!

For all planned outages, see: https://www.nersc.gov/live

For past outages, see: https://my.nersc.gov/outagelog-cs

epalmer@perlmutter:login30:~> $
epalmer@perlmutter:login30:~> $ Now what? \_(ツ)_/\
Outline: Four Main Ways of Getting Software

- Loading modules
- Containers
- Compiling from source
- Conda, Spack & E4S

Welcome to perlmutter!

For all planned outages, see: https://www.nersc.gov/live

For past outages, see: https://my.nersc.gov/outagelog.cs

epalmer@perlmutter:login30:~> $
epalmer@perlmutter:login30:~> $ Now what?  \_(ツ)_/~
Modules: Loading Preinstalled Software
**Modules For Preinstalled Software: Ex. Python3**

epalmer@perlmutter:login36:~> $ python --version

*Python 2.7.18*

epalmer@perlmutter:login36:~> $ module list

Currently Loaded Modules:

1) craype-x86-milan 6) cray-dsmml/0.2.2 11) perftools-base/23.12.0
2) libfabric/1.15.2.0 7) cray-libsci/23.12.5 12) cpe/23.12
3) craype-network-ofi 8) cray-mpich/8.1.28 13) cudatoolkit/12.2

epalmer@perlmutter:login36:~> $ module load python/3.11

epalmer@perlmutter:login36:~> $ module list

Currently Loaded Modules:

1) craype-x86-milan 6) cray-dsmml/0.2.2 11) perftools-base/23.12.0
2) libfabric/1.15.2.0 7) cray-libsci/23.12.5 12) cpe/23.12
3) craype-network-ofi 8) cray-mpich/8.1.28 13) cudatoolkit/12.2
14) python/3.11

(nersc-python) epalmer@perlmutter:login36:~> $ python --version

*Python 3.11.7*
176 Modules Currently Available on Perlmutter

Nsight-Compute: Nsight-Compute/2022.1.1
Nsight-Systems: Nsight-Systems/2022.2.1
OpenCoarrays: OpenCoarrays/2.10.1
PgEnv-acoc:
PgEnv-cray:
PgEnv-gnu:
PgEnv-intel:
PgEnv-lxvm: PgEnv-lxvm/0.1
PgEnv-nvhp:
PgEnv-nvidia:
R: R/4.2.3
amber:
acoc:
acoc-mixed:
arm-forge:
atp:erkeleyweg:
bupc: bupc/2022.10.0
bupc-narrow: bupc-narrow/2022.10.0
cce:
cce-mixed:
chapel: chapel/1.30.0
cclimate-utils:
cmake:
contrib:
common-utils: common-utils/2023q2
crona:
contrib: contrib/1.0
cpe:
cpe-cuda:
cpu: cpu/1.0
cray-R:
cray-cddb:
cray-cll:
cray-dsmi: cray-dsmi/0.2.2
cray-dyninst:
cray-fhw:
cray-hdf5:
cray-hdf5-parallel: cray-hdf5-parallel/1.12.2.9
cray-libcblas:
cray-libsci:
cray-libsci_acc: cray-libsci_acc/23.12.0
cray-lustrev: wrl-olde:
cray-mpi:
cray-mpi-abib:
cray-mpi-abib-pre-intel-5.0:
cray-mpi-abib: cray-mpi-abib/8.1.28
cray-mpi-abib: cray-mpi-abib/8.1.28
cray-mpixlate: cray-mpixlate/1.0.3
 cray-mnet:
cray-netcdf:
cray-netcdf-hdf5parallel: cray-netcdf-hdf5parallel/4.9.0.9
 cray-openshmem:
cray-pals:
cray-parallel-netcdf: cray-parallel-netcdf/1.12.3.9
 cray-pmi:
cray-python:
cray-stat:
cray-ucx:
cray-accel-amd-gfx940: craype-accel-amd-gfx940
 craype-accel-host: craype-accel-host
 craype-accel-nvidia70: craype-accel-nvidia70
 craype-accel-nvidia80: craype-accel-nvidia80
 craype-accel-nvidia90: craype-accel-nvidia90
 craype-hugepages128M: craype-hugepages128M
 craype-hugepages16M: craype-hugepages16M
 craype-hugepages1G: craype-hugepages1G
 craype-hugepages256M: craype-hugepages256M
 craype-hugepages2G: craype-hugepages2G
 craype-hugepages2M: craype-hugepages2M
 craype-hugepages32M: craype-hugepages32M
 craype-hugepages4M: craype-hugepages4M
 craype-hugepages512M: craype-hugepages512M
 craype-hugepages64M: craype-hugepages64M
 craype-network-none: craype-network-none
 craype-network-off: craype-network-off
 craype-network-ucx: craype-network-ucx
 craype-x86-milan: craype-x86-milan
 craype-x86-milan-x: craype-x86-milan-x
 craype-x86-rome: craype-x86-rome
 craype-gpu:
cudadisk:
cudnn:
darshan:
dmtoolkit:
dmtoolkit/3.0.0
dns: dns/2.15.4.5.239-2.5_48.29_g91483389
 e4s:
eigen: eigen/3.4.0
 espresso:
evp-patch: evp-patch
 fast-mkl-amd: fast-mkl-amd/fast-mkl-amd
 forge:
fpm: fpm/0.9.0
gcc: gcc-mixed:
gcc-native: gcc-native/12.3
 gcc-native-mixed: gcc-native-mixed/12.3
 gdb4hpc:
globus-tools: globus-tools/1.0
 gpp: gpp/1.0
 gnu-test: gnu-test/1.1
 gromacs:
gsl: gsl/2.7
 hip:
id:
impi: impi/2021.6.0
 intel:
intel-classic:
intel-classic-mixed:
intrinsics: intrinsics
intel-llvm: intel-llvm/2023-WW13
 intel-mixed:
intel-oneapi:
intel-oneapi-mixed:
irbu: irbu/2.0.10
jamo:
jg: jg/jg-python-jamo
julia:
lammps: lammps/2022.11.03
libfabric: libfabric/1.15.2.0
libxml2:
llvmlite: llvmlite
llvm: llvm
llm: llm
math-libs-real32: math-libs-real32/2023q2
mathematica: mathematica/13.0.1
mathlab: mathlab/2021b
matlab-mcr: matlab-mcr/2021b
mgeb: mgeb
mmp: mmp/5.5.1-gcc-11.2.0
mvasp: mvasp/5.4.4-cpu
namd:
ncc:
nersc_cr: nersc_cr/23.06
nvhpc:
nvhpc-mixed:
nvidia:
nvidia-mixed:
opencoarrays: opencoarrays/2.10.1
openmpi:
papi:
parallel: parallel/20210922
paraview: paraview/5.11.1
perftools: perftools
perftools-base:
perftools-lite:
perftools-lite-events:
perftools-lite-events:
perftools-lite-gpu:
perftools-lite-hbm:
perftools-lite-loops:
perftools-lite-loops:
perftools-preload:
petsc:
petsc/3.19.3-cpu-complex
python:
pychor:
qchem:
sanitizers4hpc:
setarg:
setarg:
spack:
spack:
spin: spin/2.0
taskrunner:
taskrunner/1.5
tensorflow:
tensorflow:
textile:
textile/2022
totalview:
training:
training/perlmutter-jan2022
upcxx:
upcxx-cuda:
upcxx-ninja:
valgrind:
valgrind4hpc:
vasp:
vasp:
vasp-tpc:
wannier90:
wrk:
wrk/4.5.2
xpmem:
xpmem/2.6.2-2.2.5_2.38_gd067c3f.shasta
### Modules Loaded at Login

#### Modules Loaded by Default:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>craype-x86-milan</td>
<td>6)</td>
</tr>
<tr>
<td>2)</td>
<td>libfabric/1.15.2.0</td>
<td>7)</td>
</tr>
<tr>
<td>3)</td>
<td>craype-network-ofi</td>
<td>8)</td>
</tr>
<tr>
<td>4)</td>
<td>xpmem/2.6.2-2.5_2.38__gd067c3f.shasta</td>
<td>9)</td>
</tr>
<tr>
<td>5)</td>
<td>PrgEnv-gnu/8.5.0</td>
<td>10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15)</td>
</tr>
</tbody>
</table>

- **CPU Architecture**
- **Default Programming Environment and Compiler**
- **GPU Architecture and CUDA-Aware MPI**
**Modules with Lmod**

**Most Common**

- module list
- module load/unload
- module swap
- module show
- module spider

**Cool Tricks**

- module --redirect -r spider . | grep <string>
- ml -t

**More information:** man module or [https://docs.nersc.gov/environment/lmod/](https://docs.nersc.gov/environment/lmod/)
Module Spider Example: Load cray-netcdf

No Longer Recommended

- module avail

*Only shows packages that can be loaded into the current module environment (due to hierarchy) – use module spider instead

More information: man module or https://docs.nersc.gov/environment/lmod/
Module Spider Example: Load cray-netcdf

No Longer Recommended

- module avail

*Only shows packages that can be loaded into the current module environment (due to hierarchy) – use module spider instead

More information: man module or https://docs.nersc.gov/environment/lmod/
Loading Modules Modifies Your Environment

epalmer@perlmutter:login25:~/Training> $ module show cray-hdf5

```
/path/cray/pe/lmod/modulefiles/compiler-gnu/12.0/cray-hdf5/1.12.2.9.lua:

family("hdf5")
conflict("PrgEnv-pathscale")
conflict("cray-hdf5")
conflict("cray-hdf5-parallel")
help([[Release info: /path/cray/pe/hdf5/1.12.2.9/release_info]])
whatis("The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing...

prepend_path("PATH","/path/cray/pe/hdf5/1.12.2.9/bin")
prepend_path("PKG_CONFIG_PATH","/path/cray/pe/hdf5/1.12.2.9/gnu/12.3/lib/pkgconfig")
prepend_path("PE_PKGCONFIG_LIBS","hdf5_hl:hdf5")
setenv("PE_HDF5_PKGCONFIG_LIBS","hdf5_hl:hdf5")
prepend_path("PE_FORTRAN_PKGCONFIG_LIBS","hdf5_hl_fortran:hdf5_fortran")
setenv("PE_HDF5_FORTRAN_PKGCONFIG_LIBS","hdf5_hl_fortran:hdf5_fortran")
prepend_path("PE_CXX_PKGCONFIG_LIBS","hdf5_hl_cpp:hdf5_cpp")
setenv("PE_HDF5_CXX_PKGCONFIG_LIBS","hdf5_hl_cpp:hdf5_cpp")
setenv("CRAY_HDF5_DIR","/path/cray/pe/hdf5/1.12.2.9")
setenv("PE_HDF5_DIR","/path/cray/pe/hdf5/1.12.2.9")
setenv("CRAY_HDF5_VERSION","1.12.2.9")
setenv("CRAY_HDF5_PREFIX","/path/cray/pe/hdf5/1.12.2.9/gnu/12.3")
setenv("HDF5_DIR","/path/cray/pe/hdf5/1.12.2.9/gnu/12.3")
setenv("HDF5_ROOT","/path/cray/pe/hdf5/1.12.2.9/gnu/12.3")
prepend_path("CRAY_LD_LIBRARY_PATH","/path/cray/pe/hdf5/1.12.2.9/gnu/12.3/lib")
prepend_path("MODULEPATH","/path/cray/pe/lmod/modulefiles/hdf5/gnu/12.0/cray-hdf5/1.12.2")
```

Path Changes
Environment Variables
Other Info
Programming Environments: Configuring Compilers and Libraries
Compiling: Your Machine vs. Perlmutter

What you may have done on another machine:

```
epalmer@home_machine:~> gcc helloworld.c -o helloworld.ex
```

- Not MPI-enabled code.

```
epalmer@home_cluster:~> mpicc helloworld.c -o helloworld.ex
```

- MPI compiler wrapper includes MPI libraries

What we recommend for Perlmutter:

```
epalmer@perlmutter:~> module load PrgEnv-gnu
epalmer@perlmutter:~> cc helloworld.c -o helloworld.ex
```

- HPE compiler wrapper includes MPI libraries, optimizations and more.
Compiler Wrappers and a Useful Option

-v and -craype-verbose will show all the inputs added by the craype (Cray Programming Environment) to the compiler when the wrappers (CC, cc, ftn) are used.

epalmer@nid005015:~/Training> **CC -craype-verbose** helloworld.cpp -o helloworld.ex

g++-12 -march=znver3 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80 -D__CRAYXT_COMPUTE_LINUX_TARGET -D__TARGET_LINUX__ helloworld.cpp -o helloworld.ex -Wl,-Bdynamic
-1/opt/cray/pe/mpich/8.1.28/ofi-gnu/12.3/include
-1/opt/cray/pe/libsci/23.12.5/GNU/12.3/x86_64/include
-1/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/nvvm/include
-1/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/CUPTI/include
-1/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/Debugger/include ...(and more)
epalmer@nid005015:~/Training> **CC -craype-verbose** helloworld.cpp -o hello

g++ -12 -march=znver3 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80 -D__CRAYXT_COMPUTE_LINUX_TARGET -D__TARGET_LINUX__ helloworld.cpp -o helloworld.ex -WI,-Bdynamic -I/opt/cray/pe/mpich/8.1.28/ofi/gnu/12.3/include -I/opt/cray/pe/libsci/23.12.5/GNU/12.3/x86_64/include -I/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/nvvm/include -I/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/CUPTI/include -I/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/Debugger/include -I/opt/cray/pe/dsmml/0.2.2/dsmml/include -I/opt/cray/xpmem/2.6.2-2.5.2.38__gd067c3f.shasta/include -L/opt/cray/pe/mpich/8.1.28/ofi/gnu/12.3/lib -L/opt/cray/pe/mpich/8.1.28/gtl/lib -L/opt/cray/pe/libsci/23.12.5/GNU/12.3/x86_64/lib -L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/lib64/stubs -L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/lib64 -L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/nvvm/lib64 -L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/CUPTI/lib64 -L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/Debugger/lib64 -L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/math_libs/12.2/lib64 -L/opt/cray/pe/dsmml/0.2.2/dsmml/lib -L/opt/cray/xpmem/2.6.2-2.5.2.38__gd067c3f.shasta/lib64 -Wl,--as-needed,-lcupti,-lcudart,-no-as-needed -lcuda -WI,--as-needed,-lsci_gnu_123_mpi,-no-as-needed -WI,--as-needed,-lmpi_gnu_123,-no-as-needed -lmpi_gtl_cuda -WI,--as-needed,-lsci_gnu_123,-no-as-needed -Idl -WI,--as-needed,-ldsmml,-no-as-needed -lxpmem -WI,--as-needed,-lgfortran,-lquadmath,-no-as-needed -WI,--as-needed,-lmvec,-no-as-needed -WI,--as-needed,-lm,-no-as-needed -WI,--as-needed,-lpthread,-no-as-needed -WI,--disable-new-dtags
Same Wrapper, Different Compiler

epalmer@perlmutter:~> **module load PrgEnv-nvidia**
epalmer@perlmutter:~> **CC -craype-verbose** helloworld.cpp -o helloworld.ex

nvc++ -tp=zen3 -acc -gpu=cc80 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80 -D__CRAYXT_COMPUTE_LINUX_TARGET -pgf90libs helloworld.cpp -o helloworld.ex -I/opt/cray/pe/mpich/8.1.28/ofi/nvidia/23.3/include …

epalmer@perlmutter:~> **module load PrgEnv-intel**
epalmer@perlmutter:~> **CC -craype-verbose** helloworld.cpp -o helloworld.ex

icpx -march=core-avx2 -mtune=core-avx2 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80 -D__CRAYXT_COMPUTE_LINUX_TARGET helloworld.cpp -o helloworld.ex -Wl,-rpath=/opt/intel/oneapi/compiler/2023.2.0/linux/compiler/lib/intel64 -l/opt/cray/pe/mpich/8.1.28/ofi/intel/2022.1/include …
<table>
<thead>
<tr>
<th>Module</th>
<th>Compiler</th>
<th>C++</th>
<th>C</th>
<th>Fortran</th>
<th>MPI Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrgEnv-gnu</td>
<td>GNU</td>
<td>CC g++</td>
<td>cc gcc</td>
<td>ftn gfortran</td>
<td>cray-mpich</td>
</tr>
<tr>
<td>PrgEnv-nvidia</td>
<td>Nvidia HPC</td>
<td>CC nvc++</td>
<td>cc nvc</td>
<td>ftn nvfortran</td>
<td>cray-mpich</td>
</tr>
<tr>
<td>PrgEnv-intel</td>
<td>Intel</td>
<td>CC icpx</td>
<td>cc icx</td>
<td>ftn ifort</td>
<td>cray-mpich</td>
</tr>
<tr>
<td>PrgEnv-cray</td>
<td>Cray</td>
<td>CC crayCC (Clang)</td>
<td>cc craycc (Clang)</td>
<td>ftn crayftn</td>
<td>cray-mpich</td>
</tr>
</tbody>
</table>

For additional compilers and more info: [https://docs.nersc.gov/development/compilers/base/](https://docs.nersc.gov/development/compilers/base/)
Automatic Links Provided By The Wrappers

• Depending on modules loaded, compiler wrappers link:
  
  MPI, LAPACK, Blas, ScaLAPACK, and more, automatically.

• Cray modules, such as cray-hdf5, cray-fftw, etc. are also linked automatically by the compiler wrappers when loaded into the user environment.

**Note:** Several scientific libraries such as, LAPACK, ScaLAPACK, Blas and QDWH, are included in cray-libsci. For more information use: man intro_libsci.
Build Systems: Autoconf and CMake

Build systems such as CMake or Autotools (Makefiles) may be coded to search for the environment variables:

- CC, for the C compiler;
- CXX, for the C++ compiler;
- FC, for the Fortran compiler.

In this case, the Cray compile wrappers can be specified with:

```
CC=$(which cc) CXX=$(which CC) FC=$(which ftn)
```

Or at the configure step,

```
./configure CC=cc CXX=CC FC=ftn F77=ftn
```

More info: [https://docs.nersc.gov/development/build-tools/autoconf-make/](https://docs.nersc.gov/development/build-tools/autoconf-make/)
[https://docs.nersc.gov/development/build-tools/cmake/](https://docs.nersc.gov/development/build-tools/cmake/)
Configure and compile the SLATE library and its tester, then install the headers and library. This will also compile BLAS++, LAPACK++, and TestSweeper.

**Option 1: Makefile**

```bash
# create make.inc file, for example:
CXX  = mpicxx    # MPI compiler wrappers recommended
FC   = mpif90
blas = openblas
CXXFLAGS = -DSLATE_HAVE_MT_BCAST

compile and install:
make && make install
```

This build will require specifying:
CC=cc  CXX=CC  FC=ftn
Build Systems Tip: CMake’s GUI, ccmake

In the CMake build directory, type: `ccmake ..`

Can confirm the correct compiler wrappers are being used by the build system.
Two Comments about Linking:

• Many modules prepend the LIBRARY_PATH so it is not necessary to specify library locations, e.g.

```
elvis@login01:~> $ module load gsl
elvis@login01:~> $ CC gsl_test.cpp -lgsl -lgslcblas -o gsl_test
```

Dynamic vs. Static Linking

• Cray wrappers build dynamically linked executables by default. (The kind that give, “error while loading shared libraries: ...” when they get misplaced at runtime.)
• On Perlmutter static compilation with -static or CRAYPE_LINK_TYPE=static can fail and is not supported.

More info: https://docs.nersc.gov/development/compilers/wrappers/
Best Practice For Compiling Code on Perlmutter:

- Use the system compiler wrappers, cc, CC, and ftn
- With build systems, verify the compiler wrappers are being used
Examples of Compiling Code
Example CPU Code Compile with MPI and OpenMP

```c
int main(int argc, char *argv[]) {
    int numprocs, rank, namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int iam = 0, np = 1;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Get_processor_name(processor_name, &namelen);

    #pragma omp parallel default(shared) private(iam, np)
    {
        np = omp_get_num_threads();
        iam = omp_get_thread_num();
        printf("Hello from thread %d out of %d from process
        %d out of %d on %s\n",
            iam, np, rank, numprocs, processor_name);
    }
    MPI_Finalize();
}
```

Hellohybrid.c – contains both MPI and OpenMP commands.

Example from: https://rcc.uchicago.edu/docs/running-jobs/hybrid/index.html
## Some Good-to-Know Compiler Settings

<table>
<thead>
<tr>
<th></th>
<th>GNU</th>
<th>Cray</th>
<th>Nvidia</th>
<th>Description/ Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>-O0</td>
<td>-O0</td>
<td>-O1</td>
<td></td>
<td>Default Optimization Level</td>
</tr>
<tr>
<td>-O0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Ofast</td>
<td>-Ofast, -flto</td>
<td>-O4, -fast</td>
<td></td>
<td>Aggressive Optimization (some may cause non-bit-identical output)</td>
</tr>
<tr>
<td>-fopenmp</td>
<td>-fopenmp</td>
<td></td>
<td>-mp (CPU), -mp=gpu (GPU offload)</td>
<td>Enable OpenMP (not default)</td>
</tr>
<tr>
<td>-g, -O0</td>
<td>-g, -O0</td>
<td>-g (-O0 by default)</td>
<td>Debug</td>
<td></td>
</tr>
<tr>
<td>-v</td>
<td>-v</td>
<td>-v</td>
<td></td>
<td>Verbose</td>
</tr>
</tbody>
</table>

For more information:

- man gcc/gfortran
- man craycc/crayftn
- man nvc/nvfortran

https://docs.nersc.gov/development/compilers/
Compile with MPI and OpenMP (CPU only)

Compile line: `cc hellohybrid.c -fopenmp -o hellohybrid`

Modules Loaded: PrgEnv-gnu
Compile with MPI and OpenMP (CPU only)

Compile line: `cc hellohybrid.c -fopenmp -o hellohybrid`

Modules Loaded: PrgEnv-gnu
Compile with MPI and OpenMP on GPU

```
epalmer@nid003676:~/NERSC_User_Training/EX1> ls
hellohybrid.c
epalmer@nid003676:~/NERSC_User_Training/EX1> module list

Currently Loaded Modules:
1) craype-x86-milan
2) libfabric/1.15.0.0
3) craype-network-ofi
4) perftools-base/22.06.0
5) xpmem/2.4.4-2.3.12.2___gff0e1d9.shasta
6) xalt/2.10.2
7) darshan/3.3.1
8) Nsight-Compute/2022.1.1
9) Nsight-Systems/2022.2.1
10) cudatoolkit/11.7
11) craype-accel-nvidia80
12) gpu/1.0
13) nvidia/22.5
14) craype/2.7.16
15) cray-dsmml/0.2.2
16) cray-mpich/8.1.17
17) cray-libsci/21.08.1.2
18) PrgEnv-nvidia/8.3.3

epalmer@nid003676:~/NERSC_User_Training/EX1> cc

Compile line: cc hellohybrid.c -mp=gpu -Minfo -o hellohybrid
```

Modules Loaded: PrgEnv-nvidia, gpu

"-Minfo" is an optional command shows which parts were converted to NVIDIA GPU code.
Compile with MPI and OpenMP on GPU

**Compile line:**

```
cc hellohybrid.c -mp=gpu -Minfo -o hellohybrid
```

**Modules Loaded:**

PrgEnv-nvidia, gpu

"-Minfo" is an optional command shows which parts were converted to NVIDIA GPU code.
## CUDA-Aware MPI

### Modules Loaded by Default:

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

### Output of `module show gpu`:

```
--------------------------
/opt/nersc/pe/modulefiles/gpu/1.0.lua:
--------------------------
family("hardware")
load("cudatoolkit")
load("craype-accel-nvidia80")
setenv("MPICH_GPU_SUPPORT_ENABLED","1")
```

**Note:** The gpu module enables support for CUDA-Aware MPI – Allowing MPI to copy data to and from GPUs. `module load cpu` will turn it off.
Compile a CUDA Code with CUDA-Aware MPI

Modules Loaded:
PrgEnv-nvidia, gpu

Note: The flag, MPICH_GPU_SUPPORT_ENABLED turns CUDA-Aware MPI on or off. Loading the gpu module, sets this flag to 1, thereby enabling the feature.
Compile a CUDA Code with CUDA-Aware MPI

Modules Loaded:
PrgEnv-nvidia, gpu

Note: The flag, MPICH_GPU_SUPPORT_ENABLED turns CUDA-Aware MPI on or off. Loading the gpu module, sets this flag to 1, thereby enabling the feature.
Compile Commands for Code with OpenACC

Modules Loaded:
  PrgEnv-nvidia, gpu

Compile with OpenACC enabled:

```
cc helloacc.c -acc -Minfo=acc -o helloacc
```

*Optional command shows which parts were converted to NVIDIA GPU code.*
Manually Specify Include, Library Location and Links

epalmer> # In this example, we will show how to manually include and link libraries during the compile step. The example code I will use, requires ☹️
Manually Specify Include, Library Location and Links
More Resources on Compiling Code

NERSC Docs:
• Compiling a code on NERSC resources:
  https://docs.nersc.gov/tutorials/playbooks/compiling/
• Compiling and Building Software:
  https://docs.nersc.gov/systems/perlmutter/#compilingbuilding-software
• Base Compilers:
  https://docs.nersc.gov/development/compilers/base/#base-compilers-on-nersc-systems
• Compiler Wrappers:
  https://docs.nersc.gov/development/compilers/wrappers/#compiler-wrappers
Supported Programming Models

and more at https://docs.nersc.gov/development/programming-models/
Where to install your code
<table>
<thead>
<tr>
<th>Location</th>
<th>Directory</th>
<th>Performance</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>$HOME</td>
<td>nonoptimal</td>
<td>Source code, scripts, small</td>
</tr>
<tr>
<td>Scratch</td>
<td>$SCRATCH</td>
<td>optimal</td>
<td>Unsharable, for temporary I/O</td>
</tr>
<tr>
<td>Common</td>
<td>/global/common/softwar/&lt;proj&gt;</td>
<td>performant</td>
<td>Optimized for software installs, Read-only on compute nodes</td>
</tr>
<tr>
<td>Community</td>
<td>$CFS/&lt;proj&gt;</td>
<td>medium</td>
<td>For sharing data among projects</td>
</tr>
</tbody>
</table>

Note: Common default install locations such as /usr are not writable for Perlmutter users.

More info: https://docs.nersc.gov/filesystems/
Spack: A Package Manager for Supercomputers
Installing a package with Spack

```
  git clone https://github.com/spack/spack.git
  . spack/share/spack/setup-env.sh
  spack install <package>
```

- Typically installs take some time as Spack will download and build many redundant dependencies.
- You can tell Spack not to do this with configuration files.

Since Spack is widely used, there is a lot of support. See https://spack.io/about/ for more information and check out their Slack channel.
Extreme-Scale Scientific Software Stack
Even More Software!
The Extreme-Scale Scientific Software Stack (E4S)

<table>
<thead>
<tr>
<th>module</th>
<th>Version</th>
<th>Command</th>
<th>Installed Packages</th>
<th>Can I Install More?</th>
</tr>
</thead>
<tbody>
<tr>
<td>e4s</td>
<td>23.08</td>
<td>module load e4s</td>
<td>478</td>
<td>Yes</td>
</tr>
</tbody>
</table>

elvis@perlmutter> module load e4s
elvis@perlmutter> spack env activate cuda
elvis@perlmutter> spack find

```bash
>>> In environment cuda
>>> Root specs
-- no arch / gcc@12.2.0
-----------------------------------------
amrex%gcc@12.2.0 +cuda cuda_arch=80
arborx%gcc@12.2.0 +cuda cuda_arch=80
caliper%gcc@12.2.0 +cuda cuda_arch=80
...
```

- E4S at NERSC is a curated software stack that get additional testing and support – delivered via the SPACK package manager

More info: https://docs.nersc.gov/applications/e4s/
## Install or Load a Spack Package

<table>
<thead>
<tr>
<th>Steps</th>
<th>Already Installed in Spack/E4S</th>
<th>Available via Spack/E4S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>spack env activate &lt;env&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Select an environment. *Needed for E4S modules, not required for Spack module</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>spack find -v</td>
<td>spack list</td>
</tr>
<tr>
<td></td>
<td>Match package config. and compiler</td>
<td>Search list of ~6700 avail. packages</td>
</tr>
<tr>
<td>3</td>
<td>spack load &lt;package&gt;</td>
<td>spack info &lt;package&gt;</td>
</tr>
<tr>
<td></td>
<td>Load desired package/variant</td>
<td>Get information about package options</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>spack install &lt;package&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installs the package into Spack</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>spack load &lt;package&gt;</td>
</tr>
</tbody>
</table>

**Bonus:** `spack load --sh <package>`

Show modifications made to your environment when a package is loaded.
Example: Install a Dependency with Spack

Suppose my example code, `gsl_test.cpp`, requires `gsl` ver. 2.7.1.
Programing Environments and Compilation

Best Practices:

- Use module spider
- Use compiler wrappers –CC, cc, and ftn– with PrgEnv modules
- Verify CMake builds with ccmake
- man files are the source of truth, i.e. man intro_mpi
- See examples builds at https://github.com/NERSC/community-software
Thanks for your Attention!

Documentation for Programming Environments and Compilation:

- Modules – https://docs.nersc.gov/environment/lmod/
- Programming Environment & Compilers - https://docs.nersc.gov/systems/perlmutter/#compilingbuilding-software
- Shell Scripts – https://docs.nersc.gov/environment/shell_startup/
- Spack – https://docs.nersc.gov/development/build-tools/spack/
- E4S – https://docs.nersc.gov/applications/e4s/perlmutter/22.05/