Using the Frontier Programming Environment

https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#programming-environment)

Togo Odbadrakh

HPC Engineer
System Acceptance & User Environment Group
Oak Ridge Leadership Computing Facility (OLCF)
Oak Ridge National Laboratory (ORNL)

LMOD Environment Modules

- Available software and libraries that differ cannot coexist simultaneously in your environment ($PATH, $LD_LIBRARY_PATH, etc).
  - `env` command will show you your complete environment.
- LMOD is used to manage build and runtime environment (https://lmod.readthedocs.io)
- Usage:

```
$ module -t list # list loaded modules
$ module avail   # Show modules that can be loaded given current env
$ module help <package>  # Help info for package (if provided)
$ module show <package>  # Show contents of module
$ module load <package> <package>…   # Add package(s) to environment
$ module unload <package> <package>… # Remove package(s) from environment
$ module reset                       # Restore system defaults
$ module restore <collection>        # Load a saved collection
$ module spider <package>            # Deep search for modules
$ module purge                       # Clear all modules from env.
```

module avail

- Shows all the available modules that *can* be loaded.
- Accepts full or partial module names.
- (D)efault and (L)oaded modules are labeled in the output.

```
$ module avail
--------------------------- /sw/frontier/modulefiles ----------------------
   DefApps/default   (L)    forge/22.1.0rc        rocm/5.1.0
   afar/14.0.0_5.0.0        forge/22.1.0          rocm/5.2.0
   afar/15.0.0_5.2.0 (D)    forge/22.1.1   (D)    rocm/5.3.0 (D)
   codee/1.6.0              rocm/4.5.2            rocm/5.4.0

  Where:
   L:  Module is loaded
   D:  Default Module

Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching
any of the "keys".
```

Directory in MODULEPATH where block of modulefiles exists. Printed in order of priority.

Any new future labels will be explained in the legend at the bottom of non-terse output.

Module Priority

- First module among duplicate package/version names in `$MODULEPATH` will be selected.

```
$ module avail conduit
 /sw/frontier/spack-envs/base/modules/spack/cray-sles15-x86_64/cray-mpich/8.1.23-d2badeh/gcc/10.3.0
   conduit/0.7.2    conduit/0.8.2    conduit/0.8.3 (D)
 /sw/frontier/spack-envs/base/modules/spack/cray-sles15-x86_64/cray-mpich/8.1.17-f42wy5g/gcc/10.3.0
   conduit/0.7.2    conduit/0.8.2    conduit/0.8.3
 /sw/frontier/spack-envs/base/modules/spack/cray-sles15-x86_64/cray-mpich/8.1.12-eg2x4ag/gcc/10.3.0
   conduit/0.7.2    conduit/0.8.2
```

- To override this behavior, alter the `$MODULEPATH`:
  - `$ module use /path/to/module/file/tree`
    - prepends the given path to `$MODULEPATH` with higher priority.
  - Can also provide the path to your own custom module tree.

module spider

- The module avail command shows what *can* be loaded *given the currently loaded modules*.
- Use module spider for deep searching of what modules are *potentially available to load*.

```
$ module -t spider kokkos/3.6.00

----------------------------------------------------------------------------------------------------
  kokkos: kokkos/3.6.00
----------------------------------------------------------------------------------------------------

    You will need to load all module(s) on any one of the lines below before the "kokkos/3.6.00" module
    is available to load.

      DefApps/default

    Help:
      Kokkos implements a programming model in C++ for writing performance
      portable applications targeting all major HPC platforms.
```

DefApps Module

- Facility-installed software are available through the module system.
  - *i.e.* ECP E4S packages and user-requested software.
- Dependent on user's loaded CPE modules (`PrgEnv-` and compiler).
- When CPE modules are changed, DefApps refreshes which facility-installed packages are available (module hierarchy at play).
- No user action needed – loaded by default.
  - Could be removed from your environment if using CPE-provided `cpe/<version>` modules.
  - To reset to a fresh start, call `$ module load Defapps`.

module spider

- Complete listing of possible modules is only reported when searching for a specific version: `module spider <package>/<version>`
- Can search using limited regular expressions:

All modules with ‘s’ in their name:
`module –t spider 's'`

```
$ module -t spider 's'
adios2/2.8.1
adios2/2.8.3
ascent/0.7.1
ascent/0.8.0
bison/3.7.6
bison/3.8.2
boost/1.79.0-cxx17
boost/1.79.0
….
```

All modules with ‘su’ in the beginning of the name:
`module –t –r spider '^su'`

```
$ module -t  -r spider '^su'
subversion/1.14.0
subversion/1.14.1
sundials/5.8.0-cpu
sundials/5.8.0
sundials/6.1.1-cpu
sundials/6.1.1
sundials/6.2.0-cpu
superlu-dist/7.1.1-cpu
superlu-dist/7.1.1
superlu-dist/7.2.0-cpu
superlu-dist/7.2.0
….
```

module show

- Once you find a module, you can see how it modifies your environment by using `module show <module>`

```
-------------------------------------------------------------------------------------------------------------------------------------------------------------
---------
   /sw/frontier/spack-envs/base/modules/spack/cray-sles15-x86_64/cce/15.0.0/kokkos/3.6.00.lua:
-------------------------------------------------------------------------------------------------------------------------------------------------------------
---------
whatis("Name : kokkos")
whatis("Version : 3.6.00")
whatis("Target : zen3")
whatis("Short description : Kokkos implements a programming model in C++ for writing performance portable applications targeting all major HPC platfor
ms.")
whatis("Configure options : -DCMAKE_POSITION_INDEPENDENT_CODE:BOOL=OFF -DCMAKE_CXX_STANDARD:STRING=14 -DBUILD_SHARED_LIBS:BOOL=ON -DKokkos_ARCH_ZEN3:B
OOL=ON -DKokkos_ARCH_VEGA90A:BOOL=ON -DKokkos_ENABLE_CUDA:BOOL=OFF -DKokkos_ENABLE_OPENMP:BOOL=OFF -DKokkos_ENABLE_PTHREAD:BOOL=OFF -DKokkos_ENABLE_SE
RIAL:BOOL=ON -DKokkos_ENABLE_HIP:BOOL=ON -DKokkos_ENABLE_SYCL:BOOL=OFF -DKokkos_ENABLE_OPENMPTARGET:BOOL=OFF -DKokkos_ENABLE_AGGRESSIVE_VECTORIZATION:
BOOL=OFF -DKokkos_ENABLE_COMPILER_WARNINGS:BOOL=OFF -DKokkos_ENABLE_CUDA_CONSTEXPR:BOOL=OFF -DKokkos_ENABLE_CUDA_LAMBDA:BOOL=OFF -DKokkos_ENABLE_CUDA_
LDG_INTRINSIC:BOOL=OFF -DKokkos_ENABLE_CUDA_RELOCATABLE_DEVICE_CODE:BOOL=OFF -DKokkos_ENABLE_CUDA_UVM:BOOL=OFF -DKokkos_ENABLE_DEBUG:BOOL=OFF -DKokkos
_ENABLE_DEBUG_BOUNDS_CHECK:BOOL=OFF -DKokkos_ENABLE_DEBUG_DUALVIEW_MODIFY_CHECK:BOOL=OFF -DKokkos_ENABLE_DEPRECATED_CODE:BOOL=OFF -DKokkos_ENABLE_EXAM
PLES:BOOL=OFF -DKokkos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=OFF -DKokkos_ENABLE_HPX_ASYNC_DISPATCH:BOOL=OFF -DKokkos_ENABLE_PROFILING:BOOL=ON -DKokkos_E
NABLE_TUNING:BOOL=OFF -DKokkos_ENABLE_PROFILING_LOAD_PRINT:BOOL=OFF -DKokkos_ENABLE_QTHREAD:BOOL=OFF -DKokkos_ENABLE_TESTS:BOOL=OFF -DKokkos_ENABLE_HP
X:BOOL=OFF -DKokkos_ENABLE_HWLOC:BOOL=OFF -DKokkos_ENABLE_NUMACTL:BOOL=OFF -DKokkos_ENABLE_MEMKIND:BOOL=OFF -DCMAKE_CXX_COMPILER:STRING=/opt/rocm-5.3.
0/bin/hipcc")
help([[Kokkos implements a programming model in C++ for writing performance
portable applications targeting all major HPC platforms.]])
prepend_path("PATH","/sw/frontier/spack-envs/base/opt/cray-sles15-zen3/cce-15.0.0/kokkos-3.6.00-coe525zpyny2uj2qzosexagb2uf6idcf/bin")
prepend_path("LD_LIBRARY_PATH","/sw/frontier/spack-envs/base/opt/cray-sles15-zen3/cce-15.0.0/kokkos-3.6.00-coe525zpyny2uj2qzosexagb2uf6idcf/lib64")
prepend_path("CMAKE_PREFIX_PATH","/sw/frontier/spack-envs/base/opt/cray-sles15-zen3/cce-15.0.0/kokkos-3.6.00-coe525zpyny2uj2qzosexagb2uf6idcf/")
prepend_path("PATH","/sw/frontier/spack-envs/base/opt/cray-sles15-zen3/cce-15.0.0/kokkos-3.6.00-coe525zpyny2uj2qzosexagb2uf6idcf/./bin")
prepend_path("CMAKE_PREFIX_PATH","/sw/frontier/spack-envs/base/opt/cray-sles15-zen3/cce-15.0.0/kokkos-3.6.00-coe525zpyny2uj2qzosexagb2uf6idcf/./")
setenv("ROCM_PATH","/opt/rocm-5.3.0")
setenv("HIP_PLATFORM","amd")
setenv("HIP_COMPILER","clang")
setenv("HIP_CLANG_PATH","/opt/rocm-5.3.0/llvm/bin")
setenv("HSA_PATH","/opt/rocm-5.3.0/hsa")
setenv("ROCMINFO_PATH","/opt/rocm-5.3.0")
setenv("DEVICE_LIB_PATH","/opt/rocm-5.3.0/amdgcn/bitcode")
setenv("HIP_DEVICE_LIB_PATH","/opt/rocm-5.3.0/amdgcn/bitcode")
setenv("HIP_PATH","/opt/rocm-5.3.0")
setenv("LLVM_PATH","/opt/rocm-5.3.0/llvm")
append_path{"HIPCC_COMPILE_FLAGS_APPEND","--rocm-path=/opt/rocm-5.3.0",delim=" "}
setenv("HCC_AMDGPU_TARGET","gfx90a")
setenv("OLCF_KOKKOS_ROOT","/sw/frontier/spack-envs/base/opt/cray-sles15-zen3/cce-15.0.0/kokkos-3.6.00-coe525zpyny2uj2qzosexagb2uf6idcf")
```

Available Compilers and Programming Environments

- PrgEnv-<toolchain> metamodules load compatible versions of the compiler, mpi, scientific/math libraries...etc.

CPE Programming Environment Module		PrgEnv-cray	PrgEnv-amd	PrgEnv-gnu	-
Host Toolchain Module		cce/17.0.0	amd/6.0.0	gcc/12.3.0	Custom User Toolchains
		cce/16.0.1(0)	amd/5.7.1(0)	gcc/11.20	
		cce/15.0.1(0)	amd/5.6.0	gcc/10.3.0	
		cce/14.0.2(1,0)	amd/5.5.1	gcc/7.5.0 (OS)	
CPE Compiler Drivers	C: cc	craycc	amdclang	gcc	-
	C++: CC	crayCC	amdclang++	g++	-
	Fortran: ftn	crayftn	amdflang	gfortran	-
ROCm Provider Module		rocm/<version>	-	rocm/<version>	rocm/<version>

- CPE provides modules amd and rocm which expose the ROCm toolchain to the environment.
- OLCF provides a rocm module for preview versions of ROCm not officially supported by the current CPE.

ROCm, Host Toolchain, and Cray MPICH Compatibility

- cray-mpich GTL (GPU Transport Layer) depends on specific version-matched ROCm runtime libraries.
- ROCm runtime libraries depend on specific LLVM runtime libraries.
- Recommended to match LLVM ABI version of LLVM-based host toolchains and ROCm library runtimes.

		ROCm Releases				
		5.7.0 (1)	5.6.0	5.5.0 (1)	5.4.0	5.3.0
LLVM ABI		LLVM 17	LLVM 16		LLVM 15	
cray-mpich Release Compatibility	8.1.28	✓	✓	✓	✓	✓
	8.1.27	✓	✓	✓	✓	✓
	8.1.26	✓	✓	✓	✓	✓
	8.1.25				✓	✓
	8.1.23 (D)				✓	✓
Host Toolchain+ROCm LLVM ABI Compatibility	cce	cce/17.0.0 rocm/5.7.0	cce/16.0.1(0) rocm/5.6.0	cce/16.0.1(0) rocm/5.5.0	cce/15.0.1(0) rocm/5.4.0	cce/15.0.1(0) rocm/5.3.0
	amd	amd/5.7.0	amd/5.6.0	amd/5.5.0	amd/5.4.0	amd/5.3.0

Typical Cray Environment

- One might use the following environment modifications to get started with a basic set up for using the Cray compiler

```
module load PrgEnv-cray
module load cce/17.0.0
module load rocm/5.7.1
module load craype-accel-amd-gfx90a
module load cmake

export MPICH_GPU_SUPPORT_ENABLED=1

export LD_LIBRARY_PATH=${CRAY_LD_LIBRARY_PATH}:${LD_LIBRARY_PATH}
```

```
~$ module -t list
craype-x86-trento
libfabric/1.15.2.0
craype-network-ofi
perftools-base/22.12.0
xpmem/2.6.2-2.5_2.22__gd067c3f.shasta
cray-pmi/6.1.8
craype/2.7.19
cray-dsmml/0.2.2
PrgEnv-cray/8.3.3
cce/17.0.0
darshan-runtime/3.4.0
hsi/default
DefApps/default
cray-libsci/23.12.5
cray-mpich/8.1.28
rocm/5.7.1
```

Managing Module Dependencies

- Conflicting modules are automatically reloaded or inactivated.

```
$ module load PrgEnv-cray

Lmod is automatically replacing "gcc/10.3.0" with "cce/15.0.0".

Lmod is automatically replacing "PrgEnv-gnu/8.3.3" with "PrgEnv-cray/8.3.3".

Due to MODULEPATH changes, the following have been reloaded:
  1) cray-mpich/8.1.23
```

- Generally, eliminates the need for modules swap <p1> <p2>
- Check stderr output for messages about deprecated modules.
- Modules generally only available when all dependencies are currently loaded.

User Collections

- Save collections of modules for easy re-use:

```
$ module save my_favorite_modules
Saved current collection of modules to: "my_favorite_modules", for system: "frontier"

$ module reset
Resetting modules to system default

$ module restore my_favorite_modules
Restoring modules from user's my_favorite_modules, for system: "frontier"

$ module savelist                # Show what collections you've saved
$ module describe <collection>   # Show modules in a collection
$ module disable <collection>    # Make a collection unrestorable (does not delete)
```

- Module file updates may break saved collections (manually load collection and re-save).
- Delete a collection: `rm ~/.lmod.d/<collection>.<system>`

Building your own Software

- Users can install their own software according to these guidelines.
- Where to install?
  - NFS filesystem is preferred given it is *not purged*.
  - Paths in /ccs/proj/<project>
- Recommend rebuilding whenever key CPE modules are updated:
  - ROCm (amd or amd-mixed modules), cray-mpich, libfabric, cray-pmi...
- Spack (https://spack.readthedocs.io/en/latest/)
  - Spack support for CPE is currently undergoing major changes and feature additions.

Thank you!

https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#programming-environment)

Togo Odbadrakh

HPC Engineer
System Acceptance & User Environment Group
Oak Ridge Leadership Computing Facility (OLCF)
Oak Ridge National Laboratory (ORNL)