The Summit Programming Environment

Intro to Summit Webinar
1 June 2018

Presenters: Matt Belhorn

ORNL is managed by UT-Battelle
for the US Department of Energy

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

LMOD Environment Modules

- Plethora of software and libraries available typically cannot coexist simultaneously in your environment.
- Build and runtime environment software and libraries managed with Lua-based LMOD (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

- The `module avail` command shows only what *can* be loaded.
- Accepts full or partial package names to limit output to matches.
- (D)efault and (L)oaded packages are indicated in output with labels.

```
$ module -w 80 avail
------------- /sw/summit/modulefiles/site/linux-rhel7-ppc64le/Core -------------
   ...
   cuda/9.1.85                          py-nose/1.3.7        (D)
   cuda/9.2.64              (D)         py-pip/9.0.1
   gcc/4.8.5                (L)         python/3.5.2
   gcc/5.4.0                            readline/6.3

  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".
```

Path in MODULEPATH where block of modulefiles exists. Typically printed in order of priority.

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

Modulefile Priority

- First modulefile among duplicate package/version names in MODULEPATH will be selected:

```
$ module -t -w 80 avail hdf5/1.10.0-patch1
/autofs/nccs-svm1_sw/summit/modulefiles/site/linux-rhel7-ppc64le/spectrum-mpi/10.2.0.0-
20180508-riohv7q/xl/20180502:
hdf5/1.10.0-patch1
/sw/summit/modulefiles/site/linux-rhel7-ppc64le/xl/20180502:
hdf5/1.10.0-patch1
```

- E.g. causes MPI-enabled builds to replace serial builds when MPI implementation is loaded.
- To override behavior, alter the MODULEPATH: `$ module use /path/to/module/file/tree`, given path is *prepended* with higher priority. Can also provide path to your own custom modulefiles.

Searching for Modules with Spider

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

```
$ module -t -w 80 spider hdf5/1.10.0-patch1
--------------------------------------------------------------------------------
  hdf5: hdf5/1.10.0-patch1
--------------------------------------------------------------------------------

    You will need to load all module(s) on any one of the lines below before the "hdf5/1.10.0-patch1" module is
available to load.
      ...
      gcc/4.8.5
      gcc/4.8.5  spectrum-mpi/10.1.0.4-20170915
      gcc/4.8.5  spectrum-mpi/10.2.0.0-20171117
      ...
      gcc/5.4.0
      gcc/5.4.0  spectrum-mpi/10.2.0.0-20171117
      ...
```

Spider (cont'd)

- Complete listing of possible modules is *only reported when searching for a specific version:* `module spider <package>/<version>`
- Can search with using limited regular expressions:
  - All modules with 'm' in their name: `module -t spider 'm'`
  - All modules starting with the letter 'm': `module -t -r spider '^m'`

Module Dependency Management

- Conflicting modulefiles are automatically reloaded or inactivated.

```
$ module load xl

Lmod is automatically replacing "gcc/4.8.5" with "xl/20180502".


Due to MODULEPATH changes, the following have been reloaded:
  1) spectrum-mpi/10.2.0.0-20180508
```

- Generally eliminates needs for `$ module swap <p1> <p2>`
- Check stderr output for messages about deprecated modules.
- Modules generally only available when all dependecies are currently loaded. Some exceptions, notably python extensions.

User Collections

- Save collection of modules for easy re-use:

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

$ module reset
Resetting modules to system default

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

$ 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)
```

- Modulefile updates may break saved collections.
  To fix: manually load desired modules, save to same name to update.
- Delete a collection: `rm ~/.lmod.d/<collection>.<system>`

Compilers

- XL, GCC, PGI, LLVM
- NVCC (via Cuda)

- New compiler versions added regularly**, especially XL.
- `char` defaults to `unsigned char` on ppc64le
  - May need to set additional compiler flags to alter behavior (see backup slides)

**For instance, the available versions and defaults described in this document may be obsolete immediately after this presentation. Check LMOD for current status.

IBM XL (Default Toolchain)

- xlc, xlC, xlc++, xlf
- Modules:
  - `xl/20180125`
  - `xl/20180502` (Default)
- Modules provide both C/C++ and fortran compilers.
  - Currently “versioned” by install date due to mixed/lockstep xlC/xlf version scheme used by IBM.
  - Will move to unified xlC/xlf version in future now that IBM has syncronised toolchain version.

IBM XL (cont'd)

- Not in `/opt/ibm` use `${OLCF_XL_ROOT}` `${OLCF_XLC_ROOT}` `${OLCF_XLF_ROOT}`
- Many wrappers exist to apply preset flags for various language standards. Thread safe option wrappers suffixed `*_r`
- See `${OLCF_XLC_ROOT}/etc/xlc.cfg.*` and `${OLCF_XLF_ROOT}/etc/xlf.cfg.*` for options enabled by various wrappers.
- Can always use the raw compilers and pass appropriate flags.
- When using MPI, default `$OMPI_FC` is `xlf90`
  - May need to change manually in your environment
  - Will be changed in future to bare `xlf`

GCC

- gcc, g++, gfortran
- Modules:
  - `gcc/4.8.5` (OS)
  - `gcc/5.4.0`,
  - `gcc/6.4.0` (default),
  - `gcc/8.1.0`
- OS compiler always in environment, ABI compatible with system libraries

PGI

- pgcc, pg++, pgfortran
- Modules:
  - `pgi/17.10-patched`
  - `pgi/18.3`
  - `pgi/18.4` (default)

LLVM

- `clang`, `xlf`
- Modules (All clang version 3.8):
  - `llvm/1.0-20171110`
  - `llvm/1.0-20171211`
  - `llvm/1.0-20180406` (Default)
- Replaces older `clang/*` modules
- Experimental, minimal support. May need to use older cuda (whichever latest release was built for) to use OpenMP offload.
- Default signed char: `-fsigned-char`

CUDA/NVCC

- Available under all compiler environments
- Modules:
  - `cuda/9.0.184`
  - `cuda/9.1.76`
  - `cuda/9.1.85`
  - `cuda/9.2.64` (Default)
- Recommend use latest, recompile on new releases.

OpenACC (Version 2.5)

- Supported only by GCC or PGI toolchains
- Flags:
  - GCC: `-fopenacc`
  - PGI: `-acc, -ta=nvidia:cc70`

OpenMP

Compiler	Enable OpenMP	3.1 Support	4.x Support	Enable OpenMP 4.X Offload
IBM	-qsmp=omp	FULL	PARTIAL	-qsmp=omp -qoffload
GCC	-fopenmp	FULL	PARTIAL	-fopenmp
PGI	-fopenmp	FULL	NONE	NONE
LLVM	-fopenmp	FULL	PARTIAL	-fopenmp -fopenmp-targets=nvptx64-nvidia-cuda --cuda-path=${OLCF_CUDA_ROOT}

MPI Implementation – IBM Spectrum-MPI

- Based on OpenMPI, similar compiler wrappers and flags
  - `mpicc`, `mpic++`, `mpiCC`, `mpifort`, `mpif77`, `mpif90`
- Modules:
  - `spectrum-mpi/10.2.0.0-20180508` (Default)
  - Older release should be avoided.
  - Updated regularly, requires recompilation.
- Launcher `jsrun` (See separate talk in this series)
- No support for alternative implementations (OpenMPI, MPICH)

Building your own Software

- Where to install?
  - NFS filesystem is preferred given it is not purged.
- Recommended to rebuild with new MPI, CUDA implementation releases
- Many packages automatically alter `$CMAKE_PREFIX_PATH, $PKG_CONFIG_PATH`
- Spack (https://spack.readthedocs.io/en/latest/)
  - Not all packages written to support ppc64le
  - Must configure to use external SMPI, CUDA, compilers.
  - Happy to share our Spack configs and settings on request.

What next?

There’s no better way to learn a new environment than to dive in.

Should you have questions or comments regarding the Summit programming environment, send them to us at `help@olcf.ornl.gov`.

We’re happy to help and incorporate your feedback.

Backup Slides

IBM XL - Options and Flags

- Code standards:
  - xlc: `-std=gnu99`, `-std=gnu11`
  - xlc++: `-std=gnu++11`, `-std=gnu++1y` (partial support)
  - xlf: `-qlanglvl=90std`, `-qlanglvl=2003std`, `-qlanglvl=2008std`
- Default signed char: `-qchar=signed`
- Define macro: `-WF,-D`

GCC – Options and Flags

- Code standards:
  - gcc: `-std=gnu99`, `-std=gnu11`
  - g++: `-std=gnu++11`, `-std=gnu++1y`
  - gfortran: `-std=f90`, `-std=f2003`, `-std=f2008`
- Default signed char: `-fsigned-char`

PGI – Options and Flags

- Code standards:
  - pgcc: `-c99`, `-c11`
  - pg++:`-std=c++11 --gnu_extensions`, `-std=c++14 --gnu_extensions`
  - Fortran code standard determined by suffix: .F90,.F03, .F08
- Default signed char: `-Mschar`

CUDA/NVCC – Options and Flags

- C++11 support: -std=c++11
- host/device `lambdas` (experimental): --expt-extended-lambda
- host/device `constexpr`s (experimental): --expt-relaxed-constexpr
- Supports XL, GCC, and PGI C++ backends:
  - Set `--ccbin` to host compiler