Using Modules to Select Packages

Last Update: 2014/09/18

Overview

The Brazos cluster uses Lmod (https://www.tacc.utexas.edu/tacc-projects/lmod) to assist users in selecting their software environment. A number of packages that have been compiled with both the GNU compilers as well as the Intel compilers. Using Modules removes the need for end users to hard code environment variables into their shell startup scripts and batch jobs.

The Lmod - User Guide (https://www.tacc.utexas.edu/tacc-projects/lmod/user-guide) is a good source of information for general Lmod usage.

At login time the Lmod facility is loaded automatically and a basic environment is loaded. This makes the `module` command available.

Modules can be loaded using the `module load` command. For example, to load the GNU GCC compiler module:

For a full list of available applications, see Modules Available on Brazos (/software/modules.html).

```
module load gcc/4.8.2
```

To view the **module** command's available options and sub-commands run:

```
module help
```

Viewing Modules

The Brazos modules are organized hierarchically based on the loaded compiler/MPI environment. A list of available modules can be viewed either by looking at modules available in the currently loaded compiler/MPI, or by looking at the modules installed on the system.

To see a list of modules currently available based on your loaded compiler/MPI, execute the `module avail` command.

To see a list of modules installed, regardless of what is currently loaded, execute the `module spider` command. The `module spider` command can also be used to search for installed modules as well as show what is required to load them.

To see a list of your currently loaded modules, execute the `module list` command.

The example below demonstrates the behavior of `module avail` when different modules are loaded. First, when no compiler is loaded, only the **Core modules** are available. The **GCC compiler dependent modules** become available once a compiler, in the example *gcc*, is loaded.

```
$ module list

Currently Loaded Modules:
  1) brazos (S)

  Where:
   (S):  Module is Sticky, requires --force to unload or purge

$ module avail

------------------------------------------ Core modules ----------------------------------
   brazos    (S)    intel/2013_sp1.3    settarg/5.7.5             totalview/8.14.0-21
   gcc/4.8.2        lmod/5.7.5          totalview/8.13.0-0  (D)

  Where:
   (S):  Module is Sticky, requires --force to unload or purge
   (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

$ module load gcc
$ module avail

---------------------------------- GCC compiler dependent modules ----------------------
   R/3.1.1             java/1.7.0_67        openblas/0.2.8         qrupdate/1.1.2
   acml/6.0.5.7        lapack/3.5.0         openblas/0.2.11 (D)    qt/4.8.6
   beagle-lib/r1260    leptonica/1.71       openmpi/1.8.1          suitesparse/4.2.1
   boost/1.56.0        mafft/7.164          openmpi/1.8.2   (D)    sysbench/0.5-r120
   eclipse/2014.1      maven/3.2.1          paml/4.8               tesseract/3.03-rc1
   ffmpeg/2.2.4        muscle/3.8.31        perl/5.18.2            velvet/1.2.10
   gnuplot/4.6.5       mvapich2/2.0         python/2.7.8    (D)
   hdf5/1.8.13         ncbi-blast/2.2.29    python/3.4.1
   icu/52.1            netcdf/4.3.2         qhull/2012.1

------------------------------------------ Core modules ----------------------------------
   brazos    (S)    intel/2013_sp1.3    settarg/5.7.5             totalview/8.14.0-21
   gcc/4.8.2        lmod/5.7.5          totalview/8.13.0-0  (D)

  Where:
   (S):  Module is Sticky, requires --force to unload or purge
   (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
```

Below is an example of searching for installed modules containing *mvapich2*. Notice that the available versions of *mvapich2* are listed.

```
$ module spider mvapich2

----------------------------------------------------------------------------
  mvapich2:
----------------------------------------------------------------------------
    Description:
      MVAPICH2

     Versions:
        mvapich2/2.0

----------------------------------------------------------------------------
  To find detailed information about mvapich2 please enter the full name.
  For example:

     $ module spider mvapich2/2.0
----------------------------------------------------------------------------
```

Below is an example of the detailed information provided by the spider command. Notice that the output lists *mvapich2/2.0* as being available by first loading either *gcc/4.8.2* or *intel/2013_sp1.3*.

```
$ module spider mvapich2/2.0

----------------------------------------------------------------------------
  mvapich2: mvapich2/2.0
----------------------------------------------------------------------------
    Description:
      MVAPICH2

    This module can only be loaded through the following modules:

      gcc/4.8.2
      intel/2013_sp1.3

    Help:
      MVAPICH2 Help message
```

Controlling Modules Loaded at Login

At login each user's environment is initially loaded with a default set of modules. The set of modules loaded upon login can be customized. During login the command `module --initial_load restore` is executed. That command system default modules which currently only contains the **brazos** module. If a user wishes to have their own personal set of modules they can create this by loading the desired modules then running `module save`

Below is an example workflow of creating and working with a module collection.

```
$ module savelist
No Named collections.
$ module load gcc openmpi
$ module list

Currently Loaded Modules:
  1) brazos (S)   2) gcc/4.8.2   3) openmpi/1.8.2

  Where:
   (S):  Module is Sticky, requires --force to unload or purge

$ module save my-mpi
Saved current collection of modules to: my-mpi
$ module savelist
Named collection list:
  1) my-mpi
$ module swap openmpi mvapich2
$ module list

Currently Loaded Modules:
  1) brazos (S)   2) gcc/4.8.2   3) mvapich2/2.0

  Where:
   (S):  Module is Sticky, requires --force to unload or purge

$ module restore my-mpi
Restoring modules to user's my-mpi
$ module list

Currently Loaded Modules:
  1) brazos (S)   2) gcc/4.8.2   3) openmpi/1.8.2

  Where:
   (S):  Module is Sticky, requires --force to unload or purge
```

In the example above, the `module savelist` shows my the saved collections. Once a set of modules was loaded, in the example *gcc* and *openmpi*, a collection was created named **my-mpi**. I then used `module swap` to swap *openmpi* with *mvapich2*. The `module restore my-mpi` command was then used to reset my loaded modules back to the modules saved for that collection.

Additional Information

Additional documentation for Lmod can be found in the Lmod User Guide (https://www.tacc.utexas.edu/tacc-projects/lmod/user-guide)