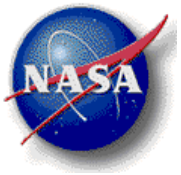
Software Applications on NCCS Systems

5/23/2024

Matt Stroud
Savannah Strong



Agenda & Objectives

- Environment modules.
- Basics of Lmod.
- Cursory overview of the applications available to users across NCCS systems.
- User installed software.

Environment Modules

- Environment modules, provide a convenient way to dynamically change the users' environment through modulefiles.
- A modulefile contains the necessary information on how to configure the environment to run a particular application or provide access to a particular library.
- Modules can be loaded and unloaded through the module system to allow for easy access to different versions of applications.

What is Lmod?

- Lmod is a Lua based module system.
- Previously we were using TCL environment modules on some of our systems.
- Knowledge of Lua is not required to interface with the module system.

Lmod Environments

Discover

- Modulefiles and installed software can be found under /usr/local/other/
- Software installed to support SLES 12 and SLES 15.

ADAPT & Prism

- Modulefiles and installed software can be found under /app (linked through /panfs/ccds02/app/)
- Software installed to support RHEL, Rocky and CentOS systems

Software Available

Type	Applications	
Scientific Programming	• R • Julia • Go • Matlab	• Octave • OpenGrADS • IDL • JDK
Visualization & Verification	• Eviz • QGIS	• GDAL • metplus
Data Processing &	• Geomatica • h4toh5tools • StereoPipeline • Esa-snap	• Globus • gsutil • gmt
Package Management	• Anaconda • Mamba	
Compilers	• gcc • Intel Compilers	• pgi • Nvhpc
Misc.	• MPI • GNU Parallel • PMIx • Apptainer (Singularity) • Podman	• Totalview • Cmake • Emacs • Nvidia SDKs

Module FAQs

- Why are there certain modules on ADAPT but not Discover? Vise versa?
  - *Modules are largely installed based on request for applications that NCCS believes would be useful to other users on the systems.*
- Why are there so many versions of certain modules?
  - *Older versions of software are retained due to other potential dependencies or compatibility issues.*
- Which version should I choose?
  - It is recommended to load the system default for a given module. This will often be the latest version that is currently available on the system.

Basic Lmod Usage

- Some basic commands needed to use Lmod include:
  - `module avail`
  - `module list`
  - `module load`
  - `module unload`
  - `module spider`
- Shorthand commands (`ml`)
- The full user guide for Lmod can be found here: https://lmod.readthedocs.io/en/latest/010_user.html

Lmod Usage Examples

$ module avail

```
---------------------------------------------------------------------------------------------------- /panfs/ccds02/app/modulefiles/core ----------------------------------------------------------------------------------------------------
  R/4.1.2                        aws/2.4.12          gcc/12.1.0       gnu_parallel/20210422       idl/8.3            nano/6.4          nvidia/12.1         (D)    totalview/2021.2.14
  R/4.3.0                 (D)    cmake/3.23.1        gdal/3.3.1       go/1.16.0                   intel/2020Update4  ncl/6.6.2         octave/7.3.0               ucx/1.12.0
  StereoPipeline/2.6.2-2019-06-17      cmake/3.24.2   (D)    geomatica/2018    go/1.18.0                  jdk/15.0.2         nco/5.0.3         opengrads/2.2.1.oga.1
  StereoPipeline/3.0.0-2021-10-12 (D)    esa-snap/9.0.0       git-lfs/3.4.0    go/1.22.1           (D)    julia/1.7.3        ncview/2.1.7      pmix/4.0.0
  anaconda/3-2022.05             (D)    gcc/9.2.0           git/2.38.0       gsutil/5.14                 llvm/10.0          ncview/2.1.8 (D)    qgis/3.18.3
  anaconda/24.3.0                       gcc/10.1.0          globus/3.2.0     gsutil/5.20         (D)    mamba/0.27.0       nvidia/11.7         singularity/3.7.3
  apache-ant/1.9.16                     gcc/11.2.0     (D)    gmt/6.2.0        h4h5tools/2.2.5             matlab/R2021b      nvidia/12.0         singularity/3.9.7    (D)

  Where:
   D:  Default Module

If the avail list is too long consider trying:

"module --default avail" or "ml -d av" to just list the default modules.
"module overview" or "ml ov" to display the number of modules for each name.

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

$ module overview

```
---------------------------------------------------------------------------------------------------- /panfs/ccds02/app/modulefiles/core ----------------------------------------------------------------------------------------------------
R           (2)  apache-ant (1)  esa-snap (1)  geomatica (1)  globus      (1)  go       (3)  idl  (1)  julia (1)  matlab (1)  nco    (1)  octave    (1)  qgis       (1)  ucx (1)
StereoPipeline (2)  aws       (1)  gcc      (4)  git-lfs  (1)  gmt         (1)  gsutil   (2)  intel (1)  llvm  (1)  nano   (1)  ncview (2)  opengrads (1)  singularity (2)
anaconda       (2)  cmake     (2)  gdal     (1)  git      (1)  gnu_parallel (1)  h4h5tools (1)  jdk   (1)  mamba (1)  ncl    (1)  nvidia (3)  pmix      (1)  totalview   (1)
```

$ ml av nvidia

```
---------------------------------------------------------------------------------------------------- /panfs/ccds02/app/modulefiles/core ----------------------------------------------------------------------------------------------------
  nvidia/11.7    nvidia/12.0    nvidia/12.1 (D)
```

…

Lmod Usage Examples

```
$ module load matlab
$ module list

Currently Loaded Modules:
  1) matlab/R2021b
```

```
$ ml anaconda
$ ml

Currently Loaded Modules:
  1) matlab/R2021b   2) anaconda/3-2022.05
```

```
$ ml -matlab gcc
$ ml

Currently Loaded Modules:
  1) anaconda/3-2022.05   2) gcc/11.2.0
```

```
$ ml purge
$ ml
No modules loaded
```

Lmod Usage Examples

```
$ ml av mpi gcc

---------------------------------------------------------------------------------------------- /usr/local/other/modulefiles/Core ----------------------------------------------------------------------------------------------
   comp/gcc/6.5.0   comp/gcc/8.3.0   comp/gcc/9.3.0    comp/gcc/10.3.0   comp/gcc/11.2.0-spack   comp/gcc/12.1.0-nvptx-omp   comp/gcc/12.1.0
   comp/gcc/7.4.0   comp/gcc/9.2.0   comp/gcc/10.1.0   comp/gcc/11.1.0   comp/gcc/11.2.0         comp/gcc/12.1.0-spack       comp/gcc/13.1.0 (D)

  Where:
   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".


$ ml comp/gcc
$ ml

Currently Loaded Modules:
  1) comp/gcc/13.1.0



$ ml av mpi

------------------------------------------------------------------------------ /usr/local/share/modulefiles/Compiler/comp/gcc/13.1.0 ------------------------------------------------------------------------------
   mpi/hpcx/2.4.0-debug       mpi/impi/18.0.5.274   mpi/impi/19.0.2.187   mpi/impi/19.1.0.166   mpi/impi/19.1.3.304   mpi/impi/2021.1.1   mpi/impi/2021.4.0 (D)   mpi/impi/2021.7.0
   mpi/hpcx/2.4.0      (D)   mpi/impi/19.0.0.117   mpi/impi/19.0.4.243   mpi/impi/19.1.1.217   mpi/impi/20.0.0.154   mpi/impi/2021.2.0   mpi/impi/2021.5.0       mpi/sgi-mpt/2.16
   mpi/impi/17.0.7.259        mpi/impi/19.0.1.144   mpi/impi/19.0.5.281   mpi/impi/19.1.2.254   mpi/impi/20.0.0.166   mpi/impi/2021.3.0   mpi/impi/2021.6.0       mpi/sgi-mpt/2.17  (D)
```

User/Group Modules

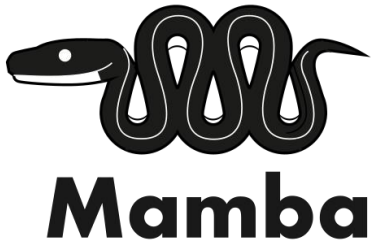
- Install in user space and create modulefiles to load in parallel with global modules.
- Modulepath will include the available system modules by default. To access user or group created modules, append the path where your module files are located to your MODULEPATH environment variable.
- For more information on creating module files see Lmod's documentation: https://lmod.readthedocs.io/en/latest/015_writing_modules.html

Other Package Management Tools

- We have a few different tools immediately available to help users install and load various software and libraries in personal and group space.
  - Anaconda
  - Mamba
  - Containers
  - Virtual Envs
- We recommend using $NOBACKUP space for installs as home directory quotas are very small.

Anaconda

- Conda allows you to install packages in environments that can be activated and deactivated much like an individual module.
- Conda is popular for creating software/library stacks for python but does support several other development languages.
- module load anaconda
- Documentation can be found here: https://docs.anaconda.com/index.html



Mamba

- Mamba is an Anaconda alternative written largely in C++ to improve performance for package resolution.
- Mamba is syntactically similar to Anaconda and can serve as a quick replacement for managing conda environments.
- module load mamba /
  module load mircomamba
- Additional documentation can be found here: https://mamba.readthedocs.io/en/latest/index.html

Containers

Containers are another way to package envs. Users build their own containers to package envs to run their applications. It's a good way to deliver a packaged env and application to a customer. The customer does not have to build an env to run the delivered application. Docker, podman, and singularity (now apptainer) are examples of tools for developing, managing, and running containers.

Here's an example of an application container. You can shell into the container and run basic unix commands and run the application. "pip freeze" shows all the package installed.

singularity shell -B /explore,/panfs,/css,/nfs4m ilab-base_6.1.0.sif

INFO: Environment variable SINGULARITY_TMPDIR is set, but APPTAINER_TMPDIR is preferred
INFO: underlay of /etc/localtime required more than 50 (107) bind mounts
Singularity> pip freeze
access==1.1.8
affine==2.4.0
amqp==5.1.1
anyio==3.6.2
argon2-cffi==21.3.0
argon2-cffi-bindings==21.2.0
arrow==1.2.3
asciitree==0.3.3
asttokens==2.2.1
async-timeout==4.0.2
attrs==22.2.0
aws-requests-auth==0.4.3
awscli==1.27.72
backcall==0.2.0
beautifulsoup4==4.11.2
billiard==3.6.4.0
.
.

Questions?

Should you have any further questions regarding current installed software or requests for installs please feel free submit a ticket to support@nccs.nasa.gov