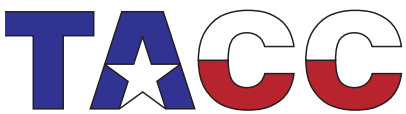
Lmod: What's New with TACC's Environment Module System

Robert McLay

The Texas Advanced Computing Center

January, 22, 2018



Introduction

- What is Lmod?
- Can Lmod help manage your site?
- Advanced Topics
- Where to go for help.

Lmod's Big Ideas

- A modern replacement for a tried and true concept.
- The guiding principal: "Make life easier w/o getting in the way."
- Reads both TCL and Lua modulefiles including Cray Modulefiles.

Fundamental Issues

- Software Packages are created and updated all the time.
- Some Users need new versions for new features and bug fixes.
- Other Users need older versions for stability and continuity.
- No system can support all versions of all packages.
- User programs using pre-built C++ & Fortran libraries must link with the same compiler.
- Similarly, MPI Applications must build and link with same MPI/Compiler pairing when using pre-built MPI libraries.

Example of Lmod: Environment Modules (I)

```
$ module list
Currently Loaded Modules:
  1) StdEnv  2) gcc/4.5  3) mpich2/1.4  4) petsc/3.1
$ module unload gcc
Inactive Modules:
  1) mpich2  2) petsc
$ module list
Currently Loaded Modules:
  1) StdEnv
Inactive Modules:
  1) mpich2  2) petsc
$ module load intel
Activating Modules:
  1) mpich2  2) petsc
$ module swap intel gcc
Due to MODULEPATH changes the follow modules have been reloaded:
  1) mpich2  2) petsc
$ module load gcc
Due to MODULEPATH changes the follow modules have been reloaded:
  1) mpich2  2) petsc
```

Example of Lmod: Environment Modules (II)

```
$  module avail
------------------ /opt/apps/modulefiles/MPI/intel/12.0/mpich2/1.4 ------------------
  petsc/3.1 (D)    petsc/3.1-debug    pmetis/4.0    tau/2.20.3

------------------- /opt/apps/modulefiles/Compiler/intel/12.0 ----------------------
  boost/1.45.0        gotoblas2/1.13      openmpi/1.4.3
  boost/1.46.0        mpich2/1.3.2        openmpi/1.5.1
  boost/1.46.1 (D)    mpich2/1.4    (D)   openmpi/1.5.3   (D)

-------------------------- /opt/apps/modulefiles/Core ------------------------------
  StdEnv                intel/11.1          papi/4.1.4
  admin/admin-1.0       intel/12.0  (D)     scite/2.28
  ddt/ddt               lmod/lmod           tex/2010
  dmalloc/dmalloc       local/local (D)     unix/unix    (D)
  fdepend/1.2           mkl/mkl             visit/visit
  gcc/4.4               noweb/2.11b
  gcc/4.5         (D)
```

Why does Lmod work at all?

- The environment is inherited from the parent process
- Changes in a child process *DOES NOT* affect the parent's environment
- So how could Lmod work at all?

The trick is:

- The `lmod` program generates text.
- The module command eval's that text.
- `module() {eval $($LMOD_CMD bash "$@";)}`
- `stdout` is evaluated
- `stderr` passes through

Why You Might Want To Use Lmod

- Same `module` command as in Tmod
- Active Development; Frequent Releases; Bug fixes.
- Vibrant Community
- It is used from Norway to Isreal to New Zealand from Stanford to MIT to NASA
- Enjoy many capabilities w/o changing a single module file
- Debian and Fedora packages available
- Many more advantages when you're ready
- It is what we use every day!

Features

- Reads for TCL and Lua modulefiles
- One name rule.
- Support Software Hierarchy (single parent only!)
- Spider Cache: fast `$ module avail`
- Properties (gpu, mic)
- Semantic Versioning: 5.6 is older than 5.10
- family("compiler") family("mpi") support
- Optional Tracking: What modules are used?
- Many other features: ml, collections, hooks, Env Vars, ...

History of Support for Module Names

- Originally only *name/version (N/V)*: gcc/4.8.1
- Lmod 5+ *cat/name/version (C/N/V)*: compiler/gcc/4.8.1
- Lmod 7+ *name/version/version (N/V/V)*: intel/impi/64/18.0.1

New with Lmod 7: N/V/V

- Support for *name/v1/v2*: fftw/64/3.3.4
- MODULERC Support:
  - Set Defaults under Site and/or User
  - Hide any installed module
- Major refactoring of Lmod
  - support N/V/V
  - Code Cleanup
  - Better Spider Cache handling

Setting Defaults

- System MODULERC file: `/path/to/lmod/etc/rc`
- `$MODULERC` points to a file.
- User `~/.modulerc`
- Can set defaults User, System, Files
- Examples: account for web services

Hiding Modules

- System MODULERC file: `/path/to/lmod/etc/rc`
- User `~/.modulerc`
- `hide-version foo/1.2.3`
- Hidden from avail, spider and keyword
- Hidden modules can be loaded
- Sites: deprecation, experimental
- show hidden: `module --show-hidden avail`

Recent Improvements

- New module function: `depends_on()`
- Reference counting on PATH like variables
- French, German, Spanish translations for Lmod messages.
- Admin list (AKA Nag List) supports Lua Regex for matching
- Improved Settarg (more on this later?)

depends_on()

- Modules X and Y depends on Module A
- ml purge; ml X; ml unload X; $\Rightarrow$ unload A
- ml purge; ml X Y; ml unload X; $\Rightarrow$ keep A
- ml purge; ml X Y; ml unload X Y ; $\Rightarrow$ unload A
- ml purge; ml A X Y; ml unload X Y ; $\Rightarrow$ keep A

Reference Counting for PATH like variables

- AKA: the /usr/local/bin problem
- Old:
  - Default path has /usr/local/bin
  - Module A also has /usr/local/bin
  - Unloading module A removes /usr/local/bin from path
- New: With Ref. Count the problem is fixed.

MODULEPATH ref counting

- A user has requested the MODULEPATH have ref-counting
- `ml unuse /path/to/modules` would always remove directory from MODULEPATH
- This is now implemented.

Future Work (I): Module Export

- Module Collections are for individuals.
- They are not meant to be shared between users
- To share I plan to add "module export"

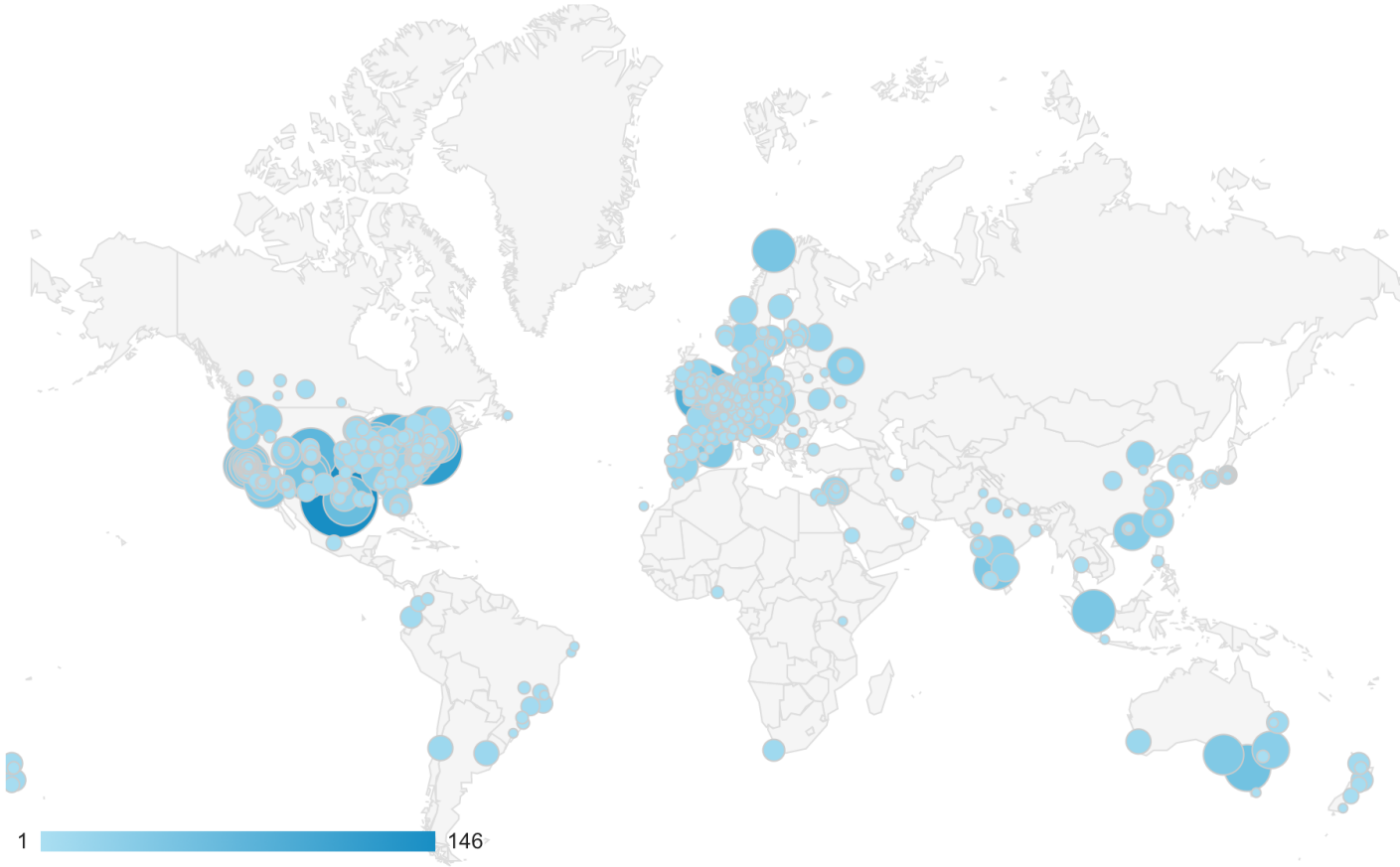
Module Export

```
$ module export <collection> > export.txt
$ cat export.txt

module purge
clearMT
export MODULEPATH=/path1:/path2

module collection_load X Y Z
module --ref_count 2 depend_on A
```

Lmod Doc usage



Can tools like Lmod improve the user experience?

- Sites provide packages: applications and libraries
- Users can pick which packages and version to suit their needs
- But what we are really after is to cut down on tickets!
- Or simply make your resources easier for your users

Lmod examples

- Lmod was first released in 2009
- It is the only module system used at TACC since 2010
- The following are some examples of how Lmod can help

Can Lmod help with the /usr/local/bin problem?

- Suppose your startup files put /usr/local/bin in PATH
- And suppose module BAR also adds /usr/local/bin to PATH
- Currently Loading then unloading BAR will remove /usr/local/bin from PATH.
- Site can configure Lmod to support duplicate paths
- Lmod now supports reference counting!

Can Lmod prevent users from mixing modules they shouldn't?

- Same Name modules:
  - Things can get confusing when users load two gcc modules
  - Normally, Lmod will unload old gcc, then load new gcc
  - Optionally, sites can auto-conflict with themselves.
- Loading two compilers or MPI Stack:
  - It is a rare user who needs to load two different compilers or two MPI stacks
  - GCC and Intel are a special case
  - Sites can add family("compiler") to compiler modules
  - This will autoswap one compiler for another!
  - Similarly for MPI modules.

How to manage software: New or Old

- How can you test new/experimental software?
- Suppose your site keeps SW for the life of machine?
- How do you encourage usage of newer SW w/o breaking old job scripts?
- Lmod now supports hiding regular modules from avail and spider.
- Hidden modules can still be loaded.
- Modules can be explicitly marked as hidden
- Or you can use the isVisible hook
- Both sites and users can hide modules

Can Lmod help with deprecating packages?

- Suppose your site keeps a limited number of versions (say 3 or less)
- How to you decide which package to keep or remove?
- Lmod support optional tracking of what packages are loaded by whom.
- You can send targeted email to those users about deprecation based on tracking.
- Independent of tracking: nag messaging
- Do not need to change modulefile!
- Users get a message when they load a deprecated module.

Can Lmod help a site that does not want default modules?

- Suppose your site produces weather forecasts or processes satellite images.
- No one set of compilers etc will satify your needs.
- Site can set LMOD_EXACT_MATCH=yes $\Rightarrow$ There are no defaults
- Users *MUST* specify name and version!

Can users have their own default list of modules?

- It is common to provide a default list of modules
- However some users will want their own modules at startup
- Users can add module commands to ~/.bashrc etc
- But this is tricky to get right.
- Lmod supports default module collections
- In fact, users can have as many named collections as they like.

Can Lmod deal with shared home filesystem?

- Suppose your site shares the home filesystem across two or more clusters
- These clusters have different modules.
- Site can set $LMOD_SYSTEM_NAME uniquely on each cluster
- This way user's collection (and personal caches) will be unique

Can users easily grep the output from Lmod?

- Lmod sends messages to `stderr` by default
- Lmod can redirect the output to `stdout` by setting $LMOD_REDIRECT=yes
- This works for bash, zsh
- It doesn't work for csh/tcsh due to the way eval works there
- Setting $LMOD_REDIRECT=yes means you lose the pager
- I do this instead: $ module –raw –redirect show impi — grep tmi

Can Lmod work with Localization and Site Messages?

- Starting Lmod 7.1+ Lmod provides the possibility of Language Translations: ES, FR, DE, and ZH
- Sites can also provide tailored message to suit their needs

Can Lmod help with software web pages?

- Many sites want to provide web pages that list the SW they provide.
- Lmod provides a tool to generate a JSON or XML list of all system modules.
- You'll have to write something to ingest the JSON or XML

Can Lmod help with compiler and/or MPI/compiler dependent modules?

- Sites can chose a Flat or Hierarchical Naming Scheme
- PETSc: A parallel iterative solver package:
  - Compilers: GCC 6.3, Intel 17.0
  - MPI Implementations: MVAPICH2 2.1, IMPI 17.0
  - MPI Solver package: PETSc 4.1
  - 4 versions of PETSc: 2 Compilers $\times$ 2 MPI
- Flat layout for PETSc
  1. `PETSc/4.1-mvapich2-2.1-gcc-6.3`
  2. `PETSc/4.1-mvapich2-2.1-intel-17.0`
  3. `PETSc/4.1-impi-17.0-gcc-6.3`
  4. `PETSc/4.1-impi-17.0-intel-17.0`

Problems w/ Flat naming scheme

- Users have to load modules:
  - "intel/17.0"
  - "mvapich2/2.1-intel-17.0"
  - "PETSc/4.1-mvapich2-2.1-intel-17.0"
  - Changing compilers means unloading all three modules
  - Reloading new compiler, MPI, PETSc modules.
  - Not loading correct modules $\Rightarrow$ Mysterious Failures!
  - Onus of package compatibility on users!
  - Or extremely complicated modulefiles!
  - Tools like EasyBuild or Spack can help here.

Hierarchical Naming Schemes

- Store modules under one tree: `/opt/apps/modulefiles`
- One strategy is to use sub-directories:
  - Core: Regular packages: apps, compilers, git
  - Compiler: Packages that depend on compiler: boost, MPI
  - MPI: Packages that depend on MPI/Compiler: PETSc, FFTW3

Loading the correct module

- User loads "`intel/17.0`" module
- Can only see/load compiler dependent packages that are built with intel 17.0 compiler.
- Can not see/load package built with other versions or other compilers.
- Similar loading "`mvapich2/2.1`" module.
- Users can only load package that are built w/ intel 17.0 and mvapich2 2.1 and no others.

Lmod works with both flat or hierarchy layouts

- Sites can chose either kind of layout.
- Lmod offers many advantages with either layout
- An Lmod site sys-admin transitioned his users by leaving the old system active
- A new hierarchy was available where all new SW was installed.
- Users can transition if/when they like.

Bash Issues

- Bash Startup is typically "broken" for non-login interactive shells
- Redhat, Centos, MacOS typically don't source /etc/bashrc on interactive shells
- MPI jobs start an interactive shell.

Bash Issues (II)

- Want module command to work in all shells.
- Want stacksize unlimited for MPI jobs
- We patched bash to force it to source /etc/tacc/bashrc

Bash Repair Choices

- Switch users to Z shell?
- patch bash (see Lmod docs)
- Expect all users to source /etc/bashrc in ~/.bashrc
- Expect all users to start jobs with #!/bin/bash -l

Debugging Lmod

- `module --config` : reports Lmod configuration
- `module -D load foo > load.log`

Tracing Lmod

- A new feature of Lmod 7.4.4+
- module -T ...
- export LMOD_TRACING=yes
- Can trace loads and how restores work.

Conclusions: Lmod 7+

- Latest version: https://github.com:TACC/Lmod.git
- Stable version: http://lmod.sf.net
- Documentation: http://lmod.readthedocs.org