Transitioning to Lmod

Robert McLay

The Texas Advanced Computing Center

January 28, 2014

Transitioning to Lmod

- What does Lmod offer?
- A transition strategy
- New features

Why You Might Want To Switch

- 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

Features requiring no changes to modulefiles

- Reads TCL modulefiles directly (Cray modules supported)
- User default and named collections of modules
- Module cache system: Faster avail, spider, etc
- Tracking module usage
- A few edge cases where Env. Modules and Lmod differ

Features of Lmod with small changes to modulefiles

- Family function: Prevent users from loading two compilers at the same time (experts can override)
- Properties: (MIC-aware, Beta, etc)
- Sticky modules
- ...

Lmod supports a software hierarchy

- Lmod supports flat layout of modules
- Some really cool features if you have a software hierarchy
  - Protecting users from mismatched modules
  - Auto Swapping of Compiler and MPI dependent modules
- When you are ready, it will be there

A Transition Strategy

- Install Lmod in your account
- Staff & Friendly Opt-in Testing
- Deploy to your users with an Opt-out choice
- Some users can run TCL/C modules
- Others can run Lmod
- No single user can run both at the same time!

Personal Testing of Lmod

- Install it in your account
  - Take current list and remove in reverse order to find required modules
  - LMOD_SYSTEM_DEFAULT_MODULES=mod1:mod2:mod3
  - **$ module purge;**
  - Redefine module command to use Lmod
  - **$ module restore;**
- I normally run a personal copy of Lmod in all my accounts (including **darter**, a Cray XC-30)
- Test, build a spider cache, module avail
- Module collections, ...

Staff & Friendly User Testing

- Install Lmod in system location.
- Install */etc/profile.d/z00_lmod.sh* to redefine the **module** cmd.
- Load system default modules (if any) after previous step
- Users who have a file named ~/`.lmod` use Lmod
- At TACC we did this for 6 months

Deploy

- When you are ready, change */etc/profile.d/z00_lmod.sh*
- Users can opt-out
- We supported this for another 6 months
- We broke Env. Module support when we added the family function
- Both transitions generated very few tickets (2 + 2)

For those who can't type: "ml"

- `ml` is a wrapper:
  - With no argument: `ml` means `module list`
  - With a module name: `ml foo` means `module load foo`.
  - With a module command: `ml spider` means `module spider`.
- See `ml --help` for more documentation.

Example of Lmod (I)

```
$  module avail
------------------ /opt/apps/modulefiles/MPI/intel/12.0/mpich2/1.4 ------------------
  petsc/3.1 (default)    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 (default)    mpich2/1.4 (default)    openmpi/1.5.3 (default)

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

Example of Lmod (II)

```
$ 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 avail vs. module spider

- Can't show every module with **module avail**
- Lmod added the command **module spider** to walk the tree.

Module Proprieties (I)

- Modules can have properties
- At TACC, we have MIC, and GPU accelerators.
- Some libraries are MIC aware.

```
add_property("arch","mic")
```

- This is controlled by the table in lmodrc.lua

Module Properties (II)

- Some modules will be "MIC" aware: mkl, fftw3, phdf5, ...
- Lmod will decorate these modules:

```
1) unix/unix     3) ddt/ddt       5) mpich2/1.5    7) phdf5/1.8.9 (m)
2) intel/13.0    4) mkl/mkl (*)   6) petsc/3.2     8) StdEnv

Where:
 (m):  module is build natively for MIC
 (*):  module is build natively for MIC and offload to the MIC

 ------
 add_property("arch","mic")              -- > phdf5
 add_property("arch","mic:offload")      -- > mkl
```

- What properties would you like to support?

Spider Cache

- Modulefiles can have properties
- Want to show properties with **module avail**
- Ugh! *Must read all modulefiles*
- Faster to read one file than walk a directory tree.
- Spider cache was created.

Spider Cache (II)

- Good News: having a spider cache speeds avail, spider
- Bad News: Sites must maintain up-to-date cache file.
- Lmod trusts valid system caches for avail and spider.
- Two kinds of caches: User & System

User spider cache file

- Used if no system cache is available.
- Written when it takes more than 2 seconds to build
- Valid for 24 hours.
- Rare if system caches exist.

System spider cache file

- Sites can have one or more cache files.
- A shared file system ⇒ One per system
- At TACC:
  - Each nodes creates a local system cache
  - Master

The Design ideas behind Lmod

- Lmod has been design to make me happy!
- It is fast, predictable and powerful.
- Action happen with minimal typing:
  - ml
  - well chosen defaults.
  - Hierarchical module layout
  - tab-completion
  - never use "**swap**"
  - don't use category/name/version.

What are the implications?

- Fast typing of commands
- Use of versions is rare.
- Strong use of autoswapping in three ways:
  - Auto swapping of the same name: ml zlib
  - Swapping of dependent modules MPI, PETSc, etc
  - Autoswapping of compilers, mpi stacks.

Numerical Software development (by me!)

- change between debug, optimize, builds
- change compilers, mpi stacks, solvers, etc
- No "make clean" required when changing any of the above
- Make it ease to compare and contrast two different builds
- Integrated with module system
- Easy to know the state

Settarg

- TARG: A family of environment variables that describe the build environment.
- Makefile can use those variables to control the build.
- Title Bar reports state of TARG
- PATH updated automatically: PATH=./$TARG:$PATH

settarg

- Provides safety, flexibility and repeatability in a dynamic environment.
- Dynamically updates the state when modules change:

```
$  env | grep '^TARG'
TARG_BUILD_SCENARIO=dbg
TARG=OBJ/_x86_64_dbg_gcc-4.6_mpich-3.0
TARG_MPI_FAMILY=mpich
TARG_MPI=mpich-3.0
$ module swap mpich openmpi; opt; env | grep '^TARG'
TARG_BUILD_SCENARIO=opt
TARG=OBJ/_x86_64_opt_gcc-4.6_openmpi-1.6
TARG_MPI=openmpi-1.6
TARG_MPI_FAMILY=openmpi
```

settarg (II)

- Typically TARG is OBJ/$ARCH_$SCENARIO_$CMPLR_$MPI
- `TARG=OBJ/_x86_64_dbg_gcc-4.6_mpich-3.0`
- User can extend this with user level or directory level specialization.
- `OBJ/_x86_64_dbg_intel-14.0_mpich-3.0_petsc-3.4`
- A makefile can modified to write generated file into $TARG.
- Never need to "`make clobber`" when switching scenario, compiler, etc.

What's new in Lmod

- Full support for reading Cray module files.
- Better support for running Lmod on shared home file systems.
- Priority Path: Some paths are more equal than others.
- **sh_to_modulefile** converts shell scripts into modulefiles.
- Support for **load(atleast("gcc","4.8"))**

Auto swap

- An improvement suggested by Maxime Boissonneault.
- Replace **module load gcc; module swap gcc intel**
- With **module load gcc; module load intel**
- Let Lmod figure out that a swap is required and do it for you!