# Lmod

Robert McLay

The Texas Advanced Computing Center

Jan. 11, 2015

## Why invent Y.A. Module System?

- Many thing right with the Original.
- But it was designed before multi-compilers/MPI
- Sites make it work but with herculean efforts
- Lmod can make this work easily.

## 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 Lua and 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 (a.k.a. Tmod)
- Others can run Lmod
- No single user can run both at the same time!
- Transistion doc: lmod.readthedocs.org

## Lmod Features

- Support for a Hierarchical Module layout
- Module spider: find all modules
- Caching system for rapid avail and spider
- Support for Properties
- Module collections, output to stdout, proper version sorting
- Reads TCL modulefiles directly (Cray modules supported)
- And so much more...

## Lmod Documentation

- lmod.readthedocs.org
- Beginning Topics: User, FAQ
- Intermediate Topics: Transitioning to Lmod, Installing Lmod, Software Hierarchy
- Advanced Topics: Generic Modules, Deprecating Modules,
- How to install Lmod on a Shared File system.
- And much more.

## Lmod handling of Cray modules

- NO DUPS: Lmod now correctly handles no duplicated in a PATH-like variable.
- TMOD_RULE: if an entry is in a path do not replace it.
- GNU4.8_LIB: Prevent Lmod from generating variables with a '.' in it.

## Injecting the Software Hierarchy into Cray Modules

- Goal: Create the Software Hierarchy where there is none.
- Create the following module /opt/apps/modulefiles/PrgEnv-intel/5.2.40.lua

```
local name = myModuleName():gsub("PrgEnv%-","")
local mpath = pathJoin("/opt/apps",name,
             myModuleVersion(),"modulefiles")
inherit()
prepend_path("MODULEPATH",mpath)
family("MPI_COMPILER")
```

- The `inherit()` function will load /opt/cray/PrgEnv-intel/5.2.40
- The directory /opt/apps/intel/5.2.40/modulefiles is prepend to MODULEPATH

## Lmod and the Pager

- Cray and Mac OS default the pager to less (not more)
- By default you have to type “q” to exit less
- Lmod now uses `LMOD_PAGER` as `less`
- `LMOD_PAGER_OPTS` as `-XqMREF`
- This will give consistant behavior across systems.

## Compiled Spider Cache

- This is work done to help filesystems like UGent.
- Kenneth wrote `update_lmod_system_cache_files`
- It build `moduleT.lua` & `dbT.lua`
- And `moduleT.luac_5.1` & `dbT.luac_5.1`
- Lmod works even when the cache is getting rebuilt.

## Optionally use cache for loads (I)

- A module load requires walking all of `$MODULEPATH`
- Lmod picks the first marked default it finds.
- If no marked default then pick the highest version.
- Lmod walks `$MODULEPATH` first then picks.
- Extending `$MODULEPATH` requires a rewalk.

## Optionally use cache for loads (II)

- By default, Lmod ignore the cache for loads.
- UGent and U. Florida want to change that.
- At UGent: module load cluster: 2.5 sec vs 1.0 sec

## Bright Computing and Lmod

- A push to get Bright to use EB and Lmod.
- Lmod has supported C/N/V or C/C/N/V since version 5.0+
- Bright uses N/A/V or N/A/A/V
- For example: intel-mpi/64/5.0.3/048, intel-mpi/mic/5.0.3/048,
- Lmod doesn't know how to tell between C/N/V and N/A/V

## What is the problem?

- In a phrase: Picking Defaults!
- ml intel-mpi
- ml intel-mpi/64
- ml intel-mpi/64/5.0.3
- These should all work!

## What is the problem? (II)

- Tmod allows for duplicate names
- For example: `module load GCC/4.8.3; module load GCC/5.2`
- Lmod allows for only one GCC
- The One-Name Rule in Lmod

## Solution

- Sites must register Architectures: 64, mic, 32, ...
- Lmod will use this to chose N/A/V over C/N/V
- If there is One A in a directory then all Dirs are A's

## Issues

- This is a major change in Lmod
- Old: UserNm either Short or Full
- New: UserNm can be inbetween Short and Full.

## Invisible Modules

- Tmod and Lmod have always supported hidden modules.
- Hidden modules are module names that start with a leading "." in the version.
- Module names with dot are ignored completely.
- They do not show up with avail or spider but can be loaded and will be listed.
- This is a way to provide a module for testing without making it publicly available
- Invisible would be new and this would be a place to discuss what this means.

## Invisible Modules: Motivation

- Cray have architecture modules that aren't needed: craype-barcelona, ...
- Sites may wish install real module names and hide them for testing
- But what does this mean?

## What are the rules?

- Can users load an invisible module? Yes
- Can you make all versions of a module invisible? Yes
- Can you make a single version invisible? Yes
- Can an invisible module be the default? No
- How will sites mark modules as invisible?
- Yet another Lua table? Combine with lmodrc.lua?

## Default Handling (I)

- The current default handling works well but ...
- It doesn't work with ˜/.modulerc when users set their own default.
- It won't work with invisible modules.

## Default handling (II)

- Selection of what the default module is already complicated.
- Especially when there are multiple directories in MODULEPATH.
- Lmod Rules for picking a default:
  - The first marked default is chosen in MODULEPATH order
  - The Highest version is chosen in MODULEPATH order.

## Default handling (III)

- Lmod current thinks it knows what the defaults are by walking the directory trees.
- The spider cache will have to be built based on walking the tree.
- But there will have to enough information to compute the default in light of ˜/.modulerc and invisible modules.

## Conclusions

- Optional support for load to use the cache.
- Support for Name/Arch/Version
- Invisible modules and what does this mean
- Default Handling