

8th Annual Lmod Booth Talk

Robert McLay

November 19, 2019

Introduction



- ▶ Features and History
- ▶ Advanced Topics
- ▶ Future work?

Features

- ▶ Reads for TCL and Lua modulefiles
- ▶ One name rule.
- ▶ Support Software Hierarchy (but not required!)
- ▶ 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 loaded?
- ▶ Many other features: ml, collections, hooks, extended default, nag ...

History of Support for Module Names

- ▶ Originally only *name/version*: gcc/4.8.1
- ▶ Lmod 5+ *cat/name/version*: compiler/gcc/4.8.1
- ▶ Lmod 7+ *name/version/version*: intel/impi/64/18.0.1

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`

Dynamic Cache files for Large Module Trees

- ▶ Groups that have a large number of specialize modules.
- ▶ Want Opt-in for these modules

Dynamic Cache files for Large Module Trees (II)

► bioPkgs.lua

```
prepend_PATH("LMOD_RC", "/path/to/cache_descript/descript.lua")  
if ( mode() ~= "spider") then  
    prepend_path("MODULEPATH", "/path/to/bioPkgs")  
end
```

► descript.lua

```
scDescript = {  
    {  
        dir = "/path/to/bioPkg/cacheDir",  
        timestamp = "/path/to/bioPkg/timestamp.txt",  
    },  
}
```

Lmod 8+ new features

- ▶ Extended Default
- ▶ The TCL interpreter is now (optionally) embedded with Lmod.
- ▶ New Function:
`extensions("numpy/1.16.4", "scipy/1.4")`
- ▶ A better way to handle special modules

Extended Default

- ▶ Long version number are a pain. (e.g. intel/18.0.4)
- ▶ With Extended Default: module load intel/18 will load the “highest” or marked default.
- ▶ Useful: Want to load intel/17 but don’t remember which is the latest 17.0.* and intel/19.0.5 is the default.

Embedded TCL interpreter

- ▶ Lmod now embeds the TCL interpreter.
- ▶ Speeds up avail and load when there are many “.version” or “.modulerc” files.
- ▶ It is still faster to use “.modulerc.lua” files over TCL version files.

extensions() function

- ▶ `extensions()`: Tells users that a module has extensions
- ▶ E.G: python has numpy and scipy
- ▶ `extensions("numpy/1.16.4", "scipy/1.4")`

extensions() function (II)


- ▶ Users can use spider to find extensions.
- ▶ Users can use avail to list extensions base name
- ▶ See examples

Special modules

- ▶ We make all modules visible including the special ones to all users.
- ▶ Some modules require a license: VASP, Matlab, etc.
- ▶ For some modules we have special groups.

Example Special modules

```
if (userInGroup("G-XYZ")) then
  prepend_path("PATH", "/opt/apps/acme_xyz/1.2.3/bin")
  -- ...
else
  color_banner("red")
  LmodMessage("You do not have access to ACME XYZ. Please see ...")
  color_banner("red")
  os.exit(1)
end
```



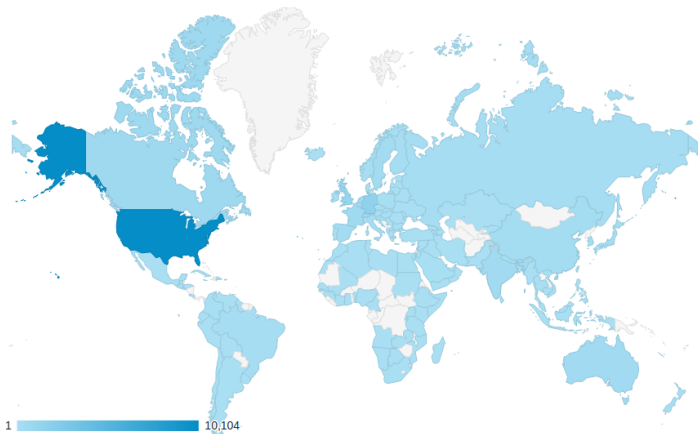
A terminal window with a black background. The first prompt is 'rios(3079)%'. The user enters 'ml acme_xyz'. The prompt changes to 'rios(3081)%'. The output shows two red horizontal lines, followed by the message 'You do not have access to ACME XYZ. Please see ...', and another two red horizontal lines.

```
rios(3079)% ml acme_xyz
You do not have access to ACME XYZ. Please see ...
rios(3081)%
```

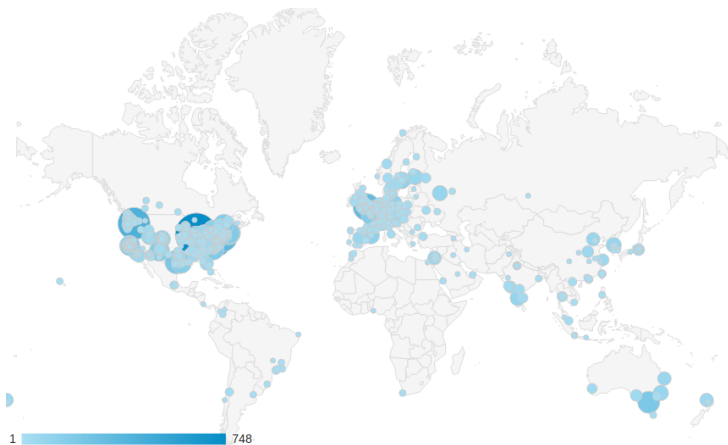
Future Work

- ▶ Lmod can optionally track usage.
- ▶ Future: Make it easier to not remember loads after 1 year.
- ▶ Get Lmod to support the break function.
- ▶ Support for Tmod4's advanced version specifiers `module load foo@2.4:2.8`
- ▶ A monthly discussion group?

Lmod Doc usage by Country



Lmod Doc usage by City



Conclusions: Lmod 8+



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