Lmod: The Lua-Based Environment Module System

Robert McLay [1] Matthew Cawood [1] Xavier Delaruelle [2]

[1]Texas Advanced Computing Center
[2] CEA

HPSF – Chicago, IL

Thanking Xavier Delaruelle

- Robert and Matthew would like to thank Xavier for presenting
- Neither of us could be at HPSF.

What is Lmod?

- Lmod is a **Lua-based environment module system** for managing software stacks on HPC systems.
- Supports both **Lua and Tcl modulefiles**.
- Enables **hierarchical MODULEPATHs**, **module families**, and **site policy enforcement**.
- Used at major HPC centers worldwide including TACC, NERSC, CSCS, and BSC.

Recap: Where We Left Off (SC24)

- Focused on improving modulefile scripting, hooks, and hierarchy handling.
- Introduced the next-generation tracking database for usage collection.
- Simplified configuration and policy definition for large sites.
- Feedback from SC24 and EUM25 presentations drove the Lmod 9 feature roadmap.

From 8.x to 9.x – What Changed Under the Hood

- Consolidated over 60 feature and performance patches into the new 9.x branch.
- New dependency engine reduces redundant file reads and improves load order resolution.
- Tracking database re-engineered – up to **100× smaller footprint** at TACC.
- Expanded site-policy features: irreversible unloads, hide/forbid logic.
- Maintains full backward compatibility with existing 8.x stacks.

New Features and Enhancements

- **Irreversible mode**: unloading a module can now set environment variables for cleanup.
- **depends_on_any()**: allows dependency on any member of a module set.
- **Enhanced hide{}/forbid{}**: expression-based control by user, group, or date (Stolen from Env. Modules)
- **Optional tracking v2**: database shrinks 100× while preserving analytics.
- **New hooks (e.g., decorate_module)** for module tagging and logging.

Performance and Reliability

- Refactored collection handling avoids NFS metadata bottlenecks.
- Improved `family()` logic speeds hierarchical resolution.
- Expanded compatibility: bash, zsh, fish, csh and nushell all supported.
- **nushell** support is new.

Performance (II)

- Discussion at EasyBuild Users Meeting EUM 25 at Jülich, Germany
- Issues with modules with large dependencies (> 50) loaded slowly
- Lmod supports **depends_on()** and **depends_on_any()** to handle module X depends on modules A B C etc
- Lmod used to re-read all currently loaded modules to report any missing dependencies on module load and unload.
- This could be slow on disk based systems.

ModuleTable

- Lmod stores its state in the user's environment
- Lmod calls this the **ModuleTable**.
- It is a Lua Table (what Python calls a **dict**)
- It is base64 encoded and is stored in env. vars **_ModuleTable001_** etc
- **module –mt** to see it decoded.

Changes to ModuleTable to track dependencies

- To remove re-reading all loaded modules
- Lmod now stores the dependencies in the ModuleTable
- As an example module A has dependencies:

```
$ module --raw show A
depends_on("x/1.0")
depends_on_any("xx","yy/1.0")
```

New ModuleTable

```
_ModuleTable_ = {
  mT = {
    A = {
      depT = {
        depA = { { sn = "x", version = {... },}, },
        doaA = { {
            { sn = "xx", version = {...}, },
            { sn = "yy", version = {...}, },
          }, }, }, },
    ...
  },
}
```

- where **depA** contains the list of **depend_on** modules
- and **doaA** contains the list of **depend_on_any** modules

Performance Improvements

- This and other changes improved module load times considerably
- Loading a module with 299 dependent modules on a Lustre filesystem
- Old: **40.2** seconds
- New **5.2** seconds

Better Error Reporting with AI agent help

- EasyBuild Community wanted better error reporting when a module was not found when using the software hierarchy.
- Lmod's spider cache is used to build the software hierarchy.
- In some cases the spider cache had an error.
- Matthew Cawood used AI agents to find bug!

review github.com/TACC/Lmod/issues/805

they provided file ./modules.tar.gz

GCCcore via depends_on. The spider cache doesn't record that. Proposing filtering the fallback to suggest only paths that work for all modules.

PROGRESS.md +5 -6

```
In the fallback branch, before adding a command for path `P`, verify that `P` works for **all** failing modules in `kB` (not just t
**Heuristic when `commonPaths` is empty (disjoint hierarchies):**

- For each candidate path `P` from module `M`:
```

PROGRESS.md is done. Here's a short summary of what went in:

PROGRESS.md Summary

Investigation

1. Extracted `modules.tar.gz` into `mtest_extract/`.
2. Module layout:
   - **Python**: `Compiler/GCCcore/12.3.0`, 13.2.0, 13.3.0
   - **QGIS**: `MPI/GCC/12.3.0/OpenMPI/4.1.5`
3. **Root cause**: `l_find_common_paths` correctly finds no common path because GCCcore and GCC/OpenMPI paths don't share a prefix.
4. The bug is in the **fallback** (lines 396–421): it suggests every path from every failing module and does not check that a path works for all modules.
5. **Spider cache**: Correct; `parentAA` matches the hierarchy. The problem is in our handling of the fallback, not the cache.

Proposed fix

- **Option B (recommended)**: When `commonPaths` is empty, only suggest paths from the module with the **fewest** paths (e.g. QGIS's single path). That yields the correct suggestion: `module load GCC/12.3.0 OpenMPI/4.1.5 Python QGIS`.

Upshot from AI Agents

- Root Cause: The bug is in the fallback: lines (396-421) in src/MainControls.lua
- Proposed fix: When **commonPaths** is empty only suggest paths from the module with the fewest paths.

Results

```
$ module load Python QGIS
Lmod has detected the following error: ...
 Try: "module spider Python QGIS" to see how to load the module(s).
   Or load any one of these options:
      module load GCC/12.3.0 OpenMPI/4.1.5 Python QGIS
```

Documentation and Usability

- Major expansion of lmod.readthedocs.io with in-depth technical docs.
- Added sections on advanced hooks, tracking v2, and policy examples.
- Clear migration guide from 8.x ⟶ 9.x.
- Real-world stats: TACC reduced usage DB size from 1 GB/day ⟶ 10 MB/day.

Site Policy and Governance

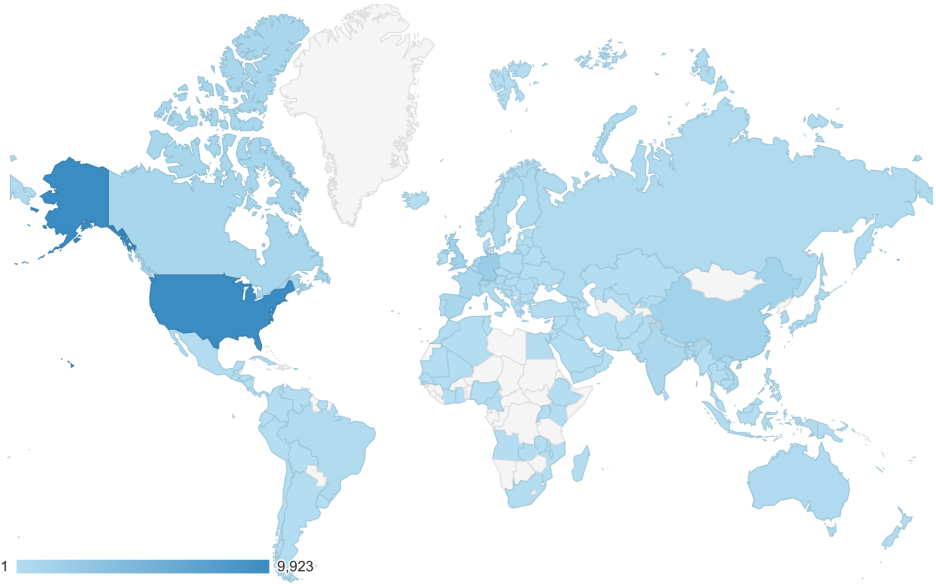
- Conditional **hide()/forbid()** rules by user, group, or time window.
- Deprecation warnings for outdated compilers or toolchains.
- Irreversible loads/unloads enable post-cleanup or variable resets.
- Simplifies compliance and software lifecycle management.

Operational Impact at TACC and Partner Sites

- On Frontera and Stampede3: module load latency **reduced by 40%** in deep hierarchies.
- Tracking database ingest reduced **100×**.
- GPU/CPU module tagging integrated via site hooks for usage analytics.
- Single `lmod_config.lua` replaced multiple legacy init scripts.

Looking Ahead

- Ongoing performance profiling for massive module trees (>100k modules).
- Additional shell and scheduler integration (Slurm, PBSPro).
- Exploring distributed tracking and federated analytics.
- Continued collaboration via GitHub and EUM workshops.


1
9,923

Get Involved

- Source: github.com/TACC/Lmod
- Docs: lmod.readthedocs.io
- Mailing lists: `lmod-announce`, `lmod-users`.
- Contribute: report issues, propose hooks, share use cases.
- This Talk: https://github.com/TACC/lmod/tree/main/my_docs/26/HPSF/presentation.pdf

Thank You

Questions? Feedback?

mclay@utexas.edu

Join Lmod Mailing list:
https://sourceforge.net/projects/lmod/lists/lmod-users