TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

Lmod: A Modern Environment Module System
XALT: Understanding HPC Usage via Job Level Collection

June 14, 2017

Outline

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

What are Modules?

- Modules help user load the SW packages they need: MPI, Boost, ...
- Modules can be unloaded.
- Modulefiles are in one file not one for each shell. (e.g. compiler init scripts)
- It is one of the main ways we communicate with our users.
- The main function of Lmod is to change the user environment
- It modifies PATH, LD_LIBRARY_PATH, ...
- It sets other environment variables and defines aliases

What is Lmod?

- A modern replacement for a tried and true concept.
- The guiding principal: “Make life easier without getting in the way.”
- Reads both TCL and Lua 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

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
- Or coming soon Lmod will support 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 a site control the output of module avail?

```
 $ module av
---- Packages compiled w/ Intel compilers -----------
   ABINIT/8.2.2            ParaView/5.3.0-OSMesa
   ARPACK-NG/3.4.0         PyMOL/1.8.6-Python-2.7.13
  [...]

------------ MPI available for Intel compilers ------
   IntelMPI/2017.2.174     ParaStationMPI/5.1.9-1 (D)
   [...]
------ Packages compiled with Intel compilers -------
   Eigen/3.3.3             Libxc/2.2.3
   [...]
--------------- Core packages ----------------------
   Advisor/2017_update2      PostgreSQL/9.6.2
   [...]

----------------- Compilers -------------------------
   GCC/5.4.0                Intel/17.2-GCC-5.4 (D)
   Intel/16.4-GCC-5.4       PGI/17.3-GCC-5.4

---------------- Recommended defaults ---------------
   defaults/CPU (D)         defaults/GPU (g)
```

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.

Lmod Features

- Reads both TCL and Lua modulefiles (mixing works fine)
- One name rule.
- Support a Software Hierarchy
- Fast `module avail` via optional spider cache
- 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, nag, ...

Conclusions: Lmod

- Latest version: https://github.com:TACC/Lmod.git
- Stable version: http://lmod.sf.net
- Documentation: http://lmod.readthedocs.org
- Shell Startup Debug: http://shellstartupdebug.sf.net
- Mailing List: lmod-users@lists.sourceforge.net.
- Join here: https://lists.sourceforge.net/lists/listinfo/lmod-users

XALT: What runs on the system

- A U.S. NSF Funded project: PI: Mark Fahey and Robert McLay
- A Census of what programs and libraries are run
- Running at TACC, NICS, U. Florida, KAUST, ...
- Integrates with TACC-Stats.

Design Goals

- Be extremely light-weight
- Provide provenance data: How?
- How many use a library or application?
- Collect Data into a Database for analysis.

Design: Linker

- XALT wraps the linker to enable tracking of exec's
- The linker (ld) wrapper intercepts the user link line.
- Generate assembly code: key-value pairs
- Capture tracemap output from ld
- Transmit collected data in *.json format

Design: Launcher

- XALT 1 used to require a wrapper for aprun, mpirun, etc
- XALT 2 no longer needs to
- Hooray! Correct wrappers were a nightmare!

Design: Transmission to DB

- File: collect nightly
- Syslog: Use Syslog filtering (or ELK)
- Direct to DB.
- Future: RabbitMQ

Lmod to XALT connection

- Lmod spider walks entire module tree.
- Can build a reverse map from paths to modules
- Can map program & libraries to modules.
- /opt/apps/i15/mv2_2_1/phdf5/1.8.14/lib/libhdf5.so.9 $\Rightarrow$ phdf5/1.8.14(intel/15.02:mvapich2/2.1)
- Also helps with function tracking.
- Tmod Sites can still use Lmod to build the reverse map.

Database Changes (I)

- Tables sizes in XALT:

Table	Size in MB
join_run_env	199603.00
join_run_object	9388.00
join_link_object	5013.00
xalt_run	4613.00
xalt_object	4175.00
xalt_link	814.00

- `join_run_env` has 2.1 billion rows

Database Changes (II)

- Environment variables are important.
- But mainly for reproducing results
- Chose a few for SQL tests.

Database Changes (III): New Design

- Store complete env $\Rightarrow$ compressed json blob
- Filter Env's with Accept Test followed by Reject Test
- Instead of 250 vars per job $\Rightarrow$ 20 to 30.
- The Filter is site controllable!

Database Changes (IV): New Design

- The “join” tables are large
- Partition “join” tables by dates or index
- Precompute views nightly.

Protecting XALT (I): UTF8 Characters

- Linux supports UTF8 Characters in file names, env. vars.
- Python supports UTF8 if you know what you are doing.
- Switch XALT to use prepared statements
- Where `query="INSERT INTO table VALUE(?,?)"`
- This prevent SQL injection: “johnny drop tables;”
- Also supports UTF8 characters.

Protecting XALT (II): Python to C++

- Difficult to protect Python from users in every case
- Solution: LD_LIBRARY_PATH="@ld_lib_path@" PATH=/usr/bin:/bin C++-exec ...
- Everything that depends on PATH must be hard coded
- basename $\Rightarrow$ /bin/basename
- Unique install for each operating system.
- Certain programs aren't in the same place: `basename`

Using XALT Data

- Targetted Outreach: Who will be affected
- Largemem Queue Overuse
- XALT and TACC-Stats

Tracking Non-mpi jobs (I)

- Originally we tracked only MPI Jobs
- By hijacking mpirun etc.
- Now we can use ELF binary format to track jobs

ELF Binary Format Trick

```
void myinit(int argc, char **argv)
{
  /* ... */
}
void myfini()
{
  /* ... */
}
  __attribute__((section(".init_array")))
       typeof(myinit) *__init = myinit;
  __attribute__((section(".fini_array")))
       typeof(myfini) *__fini = myfini;
```

Using the ELF Binary Format Trick

- This C code is compiled and linked in through the hijacked linker
- It can also be used with LD_PRELOAD
- We are using both...

Downsides

- Currently, we only track task 0 jobs.
- MPMD programs will only record the Task 0 job.
- We also lose the ability to capture return exit status

Challenges (I)

- Do not want to track mv, cp, etc
- Only want to track some executables on compute nodes
- Do not want to get overwhelmed by the data.

Answers

- XALT Tracking only when told to
- Compute node only by host name filtering
- Executable Filter based on Path
- Protection against closing stderr before fini.
- Site configurable!

Path Filtering

- Accept test, following an Ignore Test,
- Two files containing regex patterns, converted to code.
- Accept List Tests: Track /usr/bin/ddt, /bin/tar
- Ignore List Tests: /usr/bin, /bin, /sbin, ...

Using XALT 2

- A great deal of hardening
- Been running XALT 2 for 4 months with only 1 tckt.

Speeding up XALT 2

- XALT 2 generates 2 json records: at start and end
- Want to minimize measurement: Launcher jobs
- The most expensive operation is sha1sum of the shared libs
- Used to system sha1sum call in serial
- Now up-to 16 threads calls directly
- Tests show 1 sec first time 0.04 second time on Lustre.

XALT Demo

- Show modules hierarchy
- ml –raw show xalt
- Show debugging output
- type -a ld,mpirun
- Build programs
- Run tests
- Run utf8 program
- Show database results

Conclusion

- Lmod:
  - Source: github.com/TACC/lmod.git, lmod.sf.net
  - Documentation: lmod.readthedocs.org
- XALT:
  - Source: github.com/Fahey-McLay/xalt.git, xalt.sf.net
  - Documentation: doc/*.pdf, xalt.readthedocs.org