Lmod Evaluation Internal Report

2 January 2014

Version 1.7

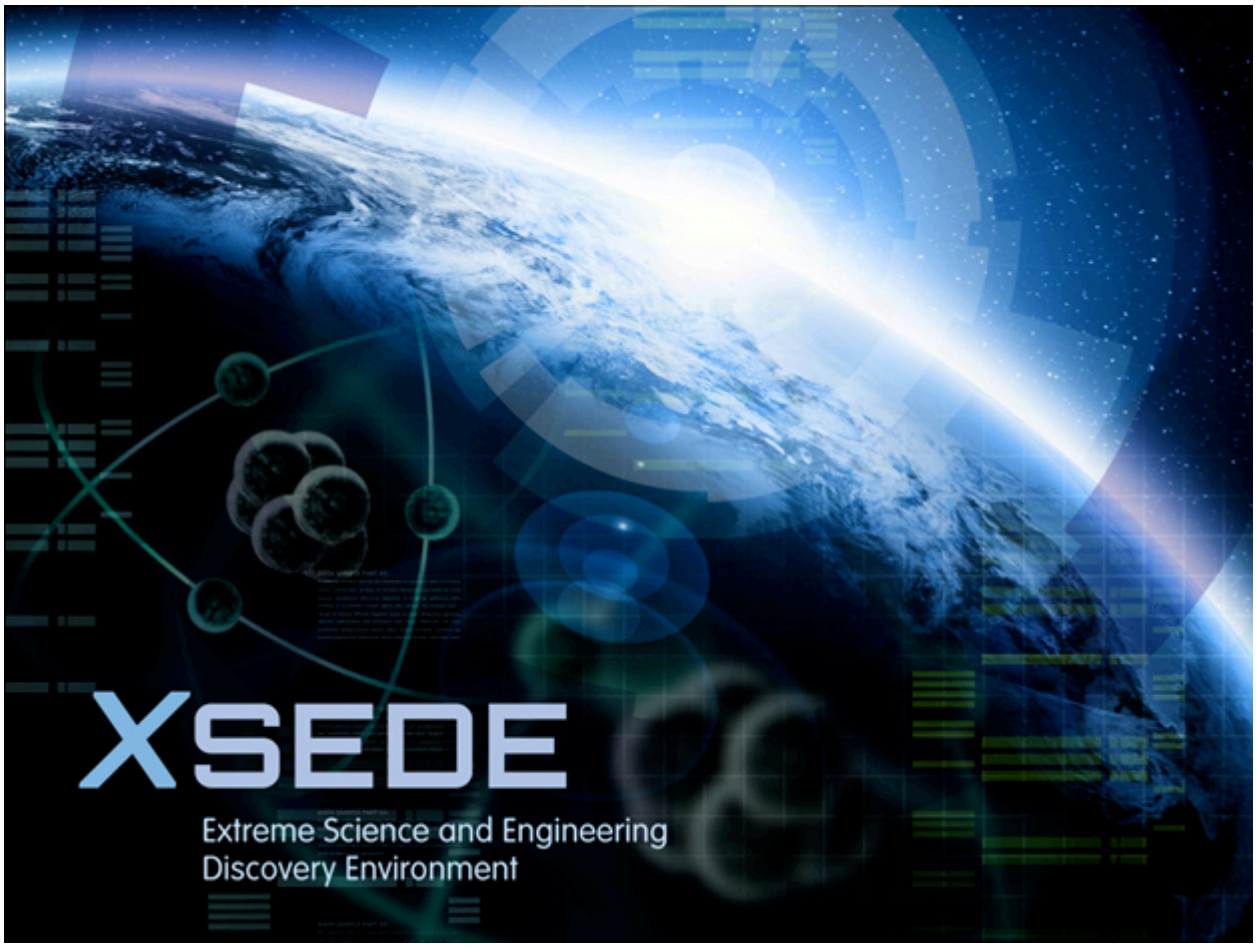

Table of Contents

Document History

Relevant Sections	Version	Date	Changes	Author
Entire Document	1	1/2/2014	Created Document	Jason Charcalla
Entire Document	1.1	1/3/2014	Edits and additions	George Butler
Entire Document	1.2	1/6/2014	Edits and additions	Jason Charcalla
Entire Document	1.3	1/6/2014	Edits and additions	George Butler
Entire Document	1.4	1/7/2014	Edits and additions	Dan Lapine
Entire Document	1.5	1/8/2014	Fixed formatting	Jason Charcalla
Prerequisites	1.6	1/14/2014	Added Lua description	Jason Charcalla
Final	1.7	1/4/2014	Final	Dan Lapine

Document Scope

This document is an internal report for the evaluation of LMOD. It is intended for internal use by the TIS group to record how the evaluation was conducted, the results of the evaluatioun, and to assist in the creation of the public report for the evaluation.

Executive Summary

An evaluation of the Lmod version 5.1.5 was performed by the Technology Investigation Service (TIS) evaluation team led by Jason Charcalla. The other members of the evaluation team were George Butler and Tony Cole. The team evaluated Lmod, an Environment Management (EM) system based on modules . Lmod is based on Lua, developed by TACC, and designed to provide additional functionality not found in other module systems. The team's testing concluded that Lmod is not only a viable alternative to other module systems currently available but provides some additional features and benefits not available in other module systems.

Lmod was chosen because of the widespread use of module-based EM systems in HPC . The ability to quickly modify ones shell environment via modules has been desirable in the community and any enhancements to current methods should be considered.

Team recommends software: YES

Introduction

Lmod is a module-based Environmental Management (EM) system designed for configuring a user's Linux or UNIX environment easily For EM systems a module is a discrete set of configurations for a specific piece of software. Lmod is written in Lua and maintained by TACC. It provides additional features over other module-based EM systems currently available. These include preventing simultaneous activation of multiple versions of the same software and a module hierarchy that guarantees packages with dependencies can only be loaded when the dependencies are also loaded.

Lmod is written in Lua and therefore should run on any Linux/UNIX host that has Lua available. The Lmod documentation states it supports multiple shells; the team found this to be true during testing.

The TIS team investigated overall functionality, additional features, and compatibility with modules written for TCL environment module systems. Additionally, the team tested Lmod on multiple Linux distributions including Centos, Suse, and Ubuntu.

Prerequisites

Lmod is written in Lua and thus requires a working Lua 5.1 or 5.2 install. Lua pronounced LOO-ah is a portable, lightweight, flexible scripting language designed to be fast and provide a procedural syntax. It is available on all UNIX, Linux, and windows platforms as well as being embeddable.

In addition to Lua the "Lua filesystem (lfs)" and "Lua posix" Lua modules are required. In this context, Lua modules are additional packages for Lua installed using the operating system's package manager. They should not be confused with modules created for Lmod. In order to access the source repository for Lmod a git client installation is needed.

Once installed the systems administrator must copy or link the provided profile files for each shell to /etc/profile.d before users will be able to take advantage of the module system.

Evaluation Details

The requirements for this evaluation were:

1. The module system must manage environment variables for software packages effectively.
2. The module system must handle conflicts and dependencies between software packages.
3. The module system must provide some means to customize each user's default environment.
4. The module system must report errors in an understandable way.
5. The module system should provide internal documentation on software packages.
6. The module system should be easy to install and maintain.
7. The module system should make the configuration for each software package easy to manage.
8. The module system must give notice that an error has occurred when it does.

The following test cases were evaluated:

1. A user can load/unload modules with ease.
2. A user can swap the currently loaded modules with new ones.
3. A user can create a custom module and load it.
4. A user can load or unloaded modules with dependencies.

Once Lua and its requisite packages are installed the process for building and installing Lmod from GIT is rather straightforward. Lmod provides profile files which must be linked or copied to /etc/profile.d in order for the module commands to function. These files required modification in cases where the Lua modules were installed in alternate locations.

The majority of our tests were preformed with prewritten module files, which are attached to the wiki page. This ensures each member of the team receives consistent results as the desired output of each test is known. These custom modules were written by the team and there for satisfy the test case where a user can create custom module files.

Lmod modules are generally written in Lua. However, the documentation states that TCL modules are supported as well. The team was successful in testing both Lua and TCL modules and included each type in its prewritten modules.

Suggestions for Evaluation Procedure Revisions

Evaluation Result

Lmod is easy to install and configure on a clean install of a system. The software enables a more powerful way to describe dependencies and conflicts between modules than what traditional TCL modules provide. Additionally, writing module files for Lmod is straight-forward since the Lua module file syntax is designed to be similar to TCL modules. Finally, Lmod reports errors clearly and exhibits no particular issues in response to scaling. In light of this, the Lmod evaluation team recommends installation of Lmod in place of TCL modules on new systems as they are deployed. Installation of Lmod on a system already using TCL modules may not be feasible. The two EM systems will not work together, and migration from TCL modules to Lmod not straightforward. The directory structure of a typical Lmod installation is necessarily more complex than that found in a TCL module installation since Lmod uses the directory structure to help implement the module hierarchy. As such, it should be noted that migrating an existing TCL module installation to Lmod on a production machine can be expected to be a major undertaking. Since the evaluation team did not attempt to perform such a migration, presenting a recommendation on whether to use Lmod in this scenario is beyond the scope of this report.

It should be noted that the team encountered unexpected results when removing or modifying the underlying files of modules currently in use. While this behavior is not recommended it is something that may occur accidentally during normal use. The team found that while Lmod did not report any errors when unloading the module the missing portions of the file (or it's entirety) were not successfully reverted on the system. This lead to a state of partially loaded modules which were not reported as being loaded at the time. Furthermore, even when module files are not modified or deleted, Lmod gives no errors upon attempting to unload a module that is not currently loaded, regardless of whether the module file exists or not. However, since this is identical to the behavior exhibited by TCL modules, it is safe to assume that it was deliberately chosen for compatibility reasons.