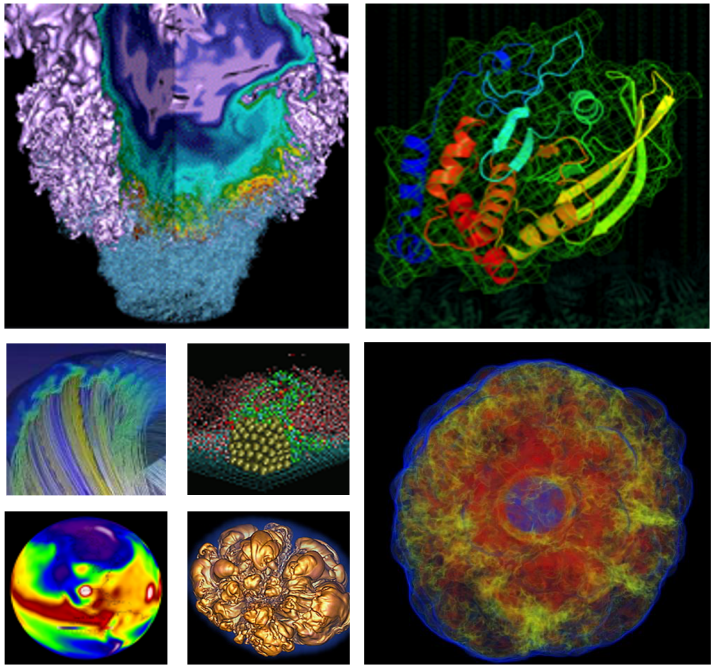
Automate Module Testing with Lmodule



Shahzeb Siddiqui
HPC Consultant/Software Integration Specialist
Lawrence Berkeley National Laboratory
6th Easybuild User Meeting
Jan 29th, 2021

https://lmodule.readthedocs.io/

https://github.com/buildtesters/lmodule

http://hpcbuildtest.slack.com/

U.S. DEPARTMENT OF ENERGY | Office of Science

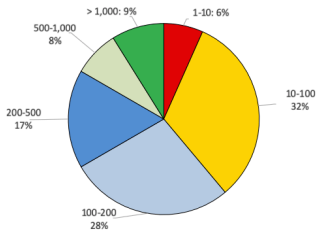
BERKELEY LAB

User Survey

- According to 3rd [Easybuild User Survey](#) (slide 30) the survey indicates 59% have more than 500+ modules deployed in production.
- Automate module testing for production stack will help increase confidence of software stack and reduce incoming user tickets

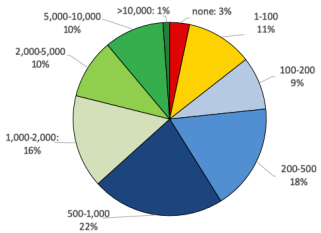
3rd easybuild User Survey

How many software installations did you perform in the last year using EasyBuild?



small shift to > 100 installations / year

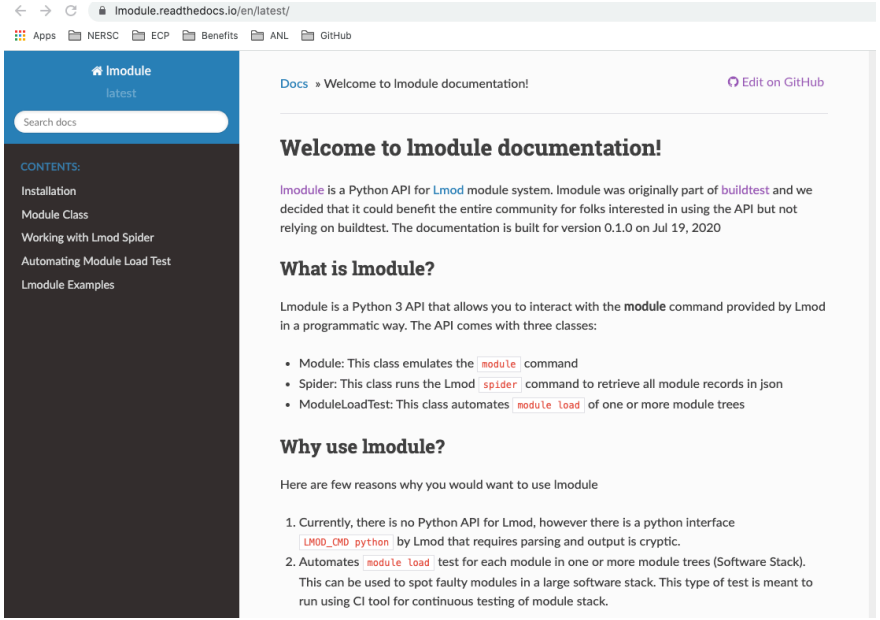
How many software installations/modules installed with EasyBuild do you currently have in production?



(new question)

What is Lmodule

- Lmodule is a Python API for module system for automating module load test for software stack
- This project grew out of buildtest and it was deprecated in v0.8.0 and now a standalone Python API available for rest of community.
- Lmodule uses spider tool from Lmod to get all modules
- To install Lmodule run **pip install lmodule** or see Installation Guide



Lmodule overview

- Currently there are three main classes in Lmodule:
  - Module – Implements the features found in module command
  - Spider – Use Lmod spider to query software stack
  - ModuleLoadTest – Automate module load test of software stack

Module

- You can pass module names as a string or list to the **Module** class and use get_command to retrieve the module command
- You can test the modules using test_modules and it will return an exit code
- You can use is_avail method to check if module is available. This runs module is-avail in backend

```
>>> from lmod.module import Module
>>> a = Module("GCCcore/8.3.0")
>>> a.get_command()
'module purge && module load GCCcore/8.3.0  '
```

```
>>> b = Module(["GCCcore/8.3.0", "zlib"])
>>> b.get_command()
'module purge && module load GCCcore/8.3.0 &&  module load zlib  '
```

```
>>> b.test_modules()
0
```

```
>>> a = Module()
>>> a.is_avail("GCC")
0

>>> a.is_avail("cuda")
1
```

Tweak Module purge behavior

- You can pass in purge and force parameters to Module class which tweaks behavior of module command. By default, we run **module purge** but this can be disabled if purge=False is passed.
- To force purge modules, pass force=True this assumes purge is enabled, but if purge=False then force will be ignored

```
>>> c = Module("OpenMPI/3.0.0", purge=False)
>>> c.get_command()
'module load OpenMPI/3.0.0  '
```

```
>>> c = Module("OpenMPI/3.0.0", force=True)
>>> c.get_command()
'module --force purge &&  module load OpenMPI/3.0.0  '
```

```
>>> c = Module("OpenMPI/3.0.0", purge=False, force=True)
>>> c.get_command()
'module load OpenMPI/3.0.0  '
```

Module Collection Support

- Lmodule supports module collections including
  - Save modules in collection (**module save**)
  - Show modules for a given collection **(module describe)**
  - Get module collection (**module restore <collection>)**
  - Test module collection
- You can use the save() method to save modules into default collection, its equivalent to running **module save**
- Alternately, you can save modules into a collection name by passing name of collection into the save method.
- In example below we save GCCcore/8.3.0 and zlib modules into collection **gcc_zlib**, in backend its loading the modules and running **module save gcc_zlib**

```
>>> b.save()
Saving modules ['GCCcore/8.3.0', 'zlib'] to module collection name: default
Saved current collection of modules to: "default"
```

```
>>> b.save("gcc_zlib")
Saving modules ['GCCcore/8.3.0', 'zlib'] to module collection name: gcc_zlib
Saved current collection of modules to: "gcc_zlib"
```

Show and Test Module Collections

- The describe method can be used to show contents of module collection. If no argument is specified it shows default collection.
- We can use get_collection to retrieve module restore command for a given module collection and use test_collection to test the collection. The test_collection will return an exit code (0, 1) which can be used to detect test faulty user collection.

```
>>> b.describe()
Collection "default" contains:
   1) GCCcore/8.3.0    2) zlib
```

```
>>> b.describe("gcc_zlib")
Collection "gcc_zlib" contains:
   1) GCCcore/8.3.0    2) zlib
```

```
>>> b.get_collection()
'module restore default'
```

```
>>> b.get_collection("gcc_zlib")
'module restore gcc_zlib'
```

```
>>> b.test_collection()
0
>>> b.test_collection("xyz")
1
```

Spider

- Lmodule can leverage Lmod spider tool to query module stack.
- Lmodule supports the following features with Spider class
  - Get unique software
  - Get Module Trees
  - Retrieve all Module Names
  - Retrieve Parent Modules
  - Get all versions of particular software

```
>>> from lmod.spider import Spider
```

Get Unique Names

- The Spider class can be called without any argument and it will search all modules in MODULEPATH.
- The get_names will retrieve unique modules reported by Spider.
- Alternately, we can filter spider output by module tree by passing root of module tree as argument to Spider class

```
>>> a = Spider()
>>> a.get_names()
['DefApps', 'adios', 'adios2', 'amgx', 'antlr', 'apr', 'apr-util', 'autoconf', 'automake', 'bison', 'blaspp', 'boost', 'bzip2', 'c-blosc', 'c
-util', 'diffutils', 'emacs', 'essl', 'expat', 'fftw', 'flex', 'font-util', 'fontconfig', 'forge', 'freetype', 'gcc', 'gdb', 'gdbm', 'gettext
spection', 'gperf', 'gsl', 'harfbuzz', 'hdf5', 'help2man', 'hpctoolkit', 'htop', 'hypre', 'ibm-wml', 'ibm-wml-ce', 'icu4c', 'inputproto', 'jd
'libcerf', 'libdwarf', 'libelf', 'libevent', 'libfabric', 'libffi', 'libgcrypt', 'libgd', 'libgpg-error', 'libiconv', 'libjpeg-turbo', 'libks
, 'libtiff', 'libtool', 'libunwind', 'libx11', 'libxau', 'libxcb', 'libxdmcp', 'libxml2', 'libxpm', 'libxslt', 'llvm', 'log4c', 'lsf-tools',
mps', 'nano', 'nasm', 'nco', 'ncurses', 'netcdf', 'netcdf-cxx4', 'netcdf-fortran', 'netlib-lapack', 'netlib-scalapack', 'npth', 'nsight-compu
pi', 'parallel-io', 'parallel-netcdf', 'parmetis', 'patchelf', 'pcre', 'perf-reports', 'perl', 'petsc', 'pgi', 'pixman', 'pkgconf', 'py-certi
py-numpy', 'py-pip', 'py-pkgconfig', 'py-pygments', 'py-setuptools', 'py-six', 'py-virtualenv', 'python', 'r', 'raja', 'rdma-core', 'readline
al', 'spectrum-mpi', 'sqlite', 'staging', 'subversion', 'superlu-dist', 'sz', 'tar', 'tau', 'tcl', 'texinfo', 'tk', 'tmux', 'udunits2', 'util
'xproto', 'xtrans', 'xz', 'zeromq', 'zfp', 'zlib', 'zstd']

>>> b = Spider("/sw/ascent/modulefiles/core")
>>> b.get_names()
['essl', 'forge', 'hpctoolkit', 'ibm-wml', 'ibm-wml-ce', 'job-step-viewer', 'perf-reports', 'python', 'scalasca', 'scorep', 'staging', 'tau']
```

Filter modules with Spider

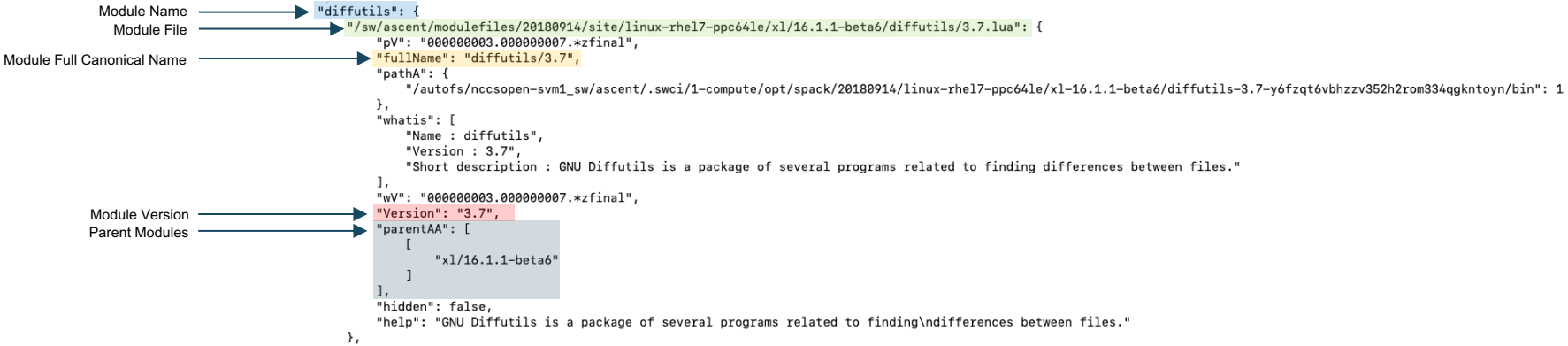
- The get_modules can be used to filter modules by name, this can be used to test a subset of modules.
- We can pass a list of module names to get_modules which will filter out all modules except for ones provided. This can used to find all modules for a given name
- In this example we use Spider class to retrieve all modules and filter by cuda and gcc modules then we test each module

```
from lmod.spider import Spider
from lmod.module import Module

a = Spider()

for module in a.get_modules(["cuda","gcc"]).values():
  print(f"module: {module}")
  cmd = Module(module,debug=True)
  cmd.test_modules()
```

```
[shahzebsiddiqui@login1.ascent ~]$ python get_names.py
module: cuda/9.1.85
[DEBUG] Executing module command: module purge && module load cuda/9.1.85
[DEBUG] Return Code: 0
module: cuda/11.0.1
[DEBUG] Executing module command: module purge && module load cuda/11.0.1
[DEBUG] Return Code: 0
module: cuda/10.1.105
[DEBUG] Executing module command: module purge && module load cuda/10.1.105
[DEBUG] Return Code: 0
module: cuda/11.0.2
[DEBUG] Executing module command: module purge && module load cuda/11.0.2
[DEBUG] Return Code: 0
module: cuda/10.1.243
[DEBUG] Executing module command: module purge && module load cuda/10.1.243
[DEBUG] Return Code: 0
module: cuda/9.2.148
[DEBUG] Executing module command: module purge && module load cuda/9.2.148
[DEBUG] Return Code: 0
module: gcc/4.8.5
[DEBUG] Executing module command: module purge && module load gcc/4.8.5
[DEBUG] Return Code: 0
module: gcc/6.4.0
[DEBUG] Executing module command: module purge && module load gcc/6.4.0
[DEBUG] Return Code: 0
module: gcc/8.1.0
[DEBUG] Executing module command: module purge && module load gcc/8.1.0
[DEBUG] Return Code: 0
module: gcc/10.1.0
[DEBUG] Executing module command: module purge && module load gcc/10.1.0
[DEBUG] Return Code: 0
module: gcc/7.4.0
[DEBUG] Executing module command: module purge && module load gcc/7.4.0
[DEBUG] Return Code: 0
module: gcc/8.1.1
[DEBUG] Executing module command: module purge && module load gcc/8.1.1
[DEBUG] Return Code: 0
module: gcc/5.4.0
[DEBUG] Executing module command: module purge && module load gcc/5.4.0
[DEBUG] Return Code: 0
```

Spider Content

- The Spider class runs $LMOD_DIR/spider –o spider-json $MODULEPATH and loads the JSON content and parses the structure to extract meaningful data
- The spider content has changed between Lmod version 6 and 7. This is based on Lmod 8.2.10 on Ascent



ModuleLoadTest

- The ModuleLoadTest class is only compatible with Lmod since it relies on spider which aims to automate module load test
- The current features includes:
  - Test all modules by MODULEPATH (default behavior)
  - Test all modules by specific tree
  - Tweak module purge behavior during test
  - Filter modules by name and full canonical name
  - Set threshold to stop after X number of tests
  - Test modules in login shell

ModuleLoadTest

- To get started you can import the class via

```
>>> from lmod.moduleloadtest import ModuleLoadTest
```

- To test all modules set by MODULEPATH just call the class

```
>>> a = ModuleLoadTest()
Testing the Following Module Trees: /sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core:/sw/ascent/modulefiles/core
________________________________________________________________________________
PASSED -  Module Name: DefApps ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/DefApps.lua )
FAILED -  Module Name: adios/1.11.1-py2 ( modulefile=/autofs/nccsopen-svm1_sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/spectrum-mpi/10.2.0.7-20180830-ttusany/gcc/8.1.1/adios/1.11.1-py2.lua )
FAILED -  Module Name: adios/1.11.1-py2 ( modulefile=/autofs/nccsopen-svm1_sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/spectrum-mpi/10.2.0.7-20180830-ttusany/gcc/8.1.1/adios/1.11.1-py2.lua )
FAILED -  Module Name: adios/1.11.1-py2 ( modulefile=/autofs/nccsopen-svm1_sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/spectrum-mpi/10.2.0.7-20180830-ttusany/gcc/8.1.1/adios/1.11.1-py2.lua )
FAILED -  Module Name: adios/1.13.1-py2 ( modulefile=/autofs/nccsopen-svm1_sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/spectrum-mpi/10.3.1.2-20200121-tvzx7uj/gcc/4.8.5/adios/1.13.1-py2.lua )
```

Tweak Purge Behavior

- We can tweak purge behavior during module test using purge=[True|False]
- The debug parameter will show the command executed along with return code of test. By default debug is disabled

```
>>> a = ModuleLoadTest("/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core",name=["cuda"],purge=True,debug=True)
Testing the Following Module Trees: /sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core
------------------------------------------------------------------------------
[DEBUG] Executing module command: module purge && module load cuda/9.1.85
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/9.1.85 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/9.1.85 )
[DEBUG] Executing module command: module purge && module load cuda/11.0.1
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/11.0.1 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/11.0.1 )
[DEBUG] Executing module command: module purge && module load cuda/10.1.105
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/10.1.105 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/10.1.105 )
[DEBUG] Executing module command: module purge && module load cuda/11.0.2
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/11.0.2 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/11.0.2 )
[DEBUG] Executing module command: module purge && module load cuda/10.1.243
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/10.1.243 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/10.1.243 )
[DEBUG] Executing module command: module purge && module load cuda/9.2.148
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/9.2.148 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/9.2.148 )
```

```
>>> a = ModuleLoadTest("/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core",name=["cuda"],purge=False,debug=True)
Testing the Following Module Trees: /sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core
------------------------------------------------------------------------------
[DEBUG] Executing module command: module load cuda/9.1.85
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/9.1.85 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/9.1.85 )
[DEBUG] Executing module command: module load cuda/11.0.1
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/11.0.1 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/11.0.1 )
[DEBUG] Executing module command: module load cuda/10.1.105
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/10.1.105 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/10.1.105 )
[DEBUG] Executing module command: module load cuda/11.0.2
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/11.0.2 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/11.0.2 )
[DEBUG] Executing module command: module load cuda/10.1.243
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/10.1.243 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/10.1.243 )
[DEBUG] Executing module command: module load cuda/9.2.148
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/9.2.148 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/9.2.148 )
```

Test modules in Login Shell

- By default all modules are tested in subshell, however we can test each module in login shell using the login=True
- This can be useful when testing modules in clean environment.

```
>>> a = ModuleLoadTest("/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core",name=["cuda"],login=True,purge=True,debug=True)
Testing the Following Module Trees: /sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core
________________________________________________________________________
[DEBUG] Executing module command: bash -l -c "module purge && module load cuda/9.1.85  "
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/9.1.85 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/9.1.85 )
[DEBUG] Executing module command: bash -l -c "module purge && module load cuda/11.0.1  "
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/11.0.1 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/11.0.1 )
[DEBUG] Executing module command: bash -l -c "module purge && module load cuda/10.1.105  "
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/10.1.105 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/10.1.105 )
[DEBUG] Executing module command: bash -l -c "module purge && module load cuda/11.0.2  "
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/11.0.2 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/11.0.2 )
[DEBUG] Executing module command: bash -l -c "module purge && module load cuda/10.1.243  "
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/10.1.243 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/10.1.243 )
[DEBUG] Executing module command: bash -l -c "module purge && module load cuda/9.2.148  "
[DEBUG] Return Code: 0
PASSED -  Module Name: cuda/9.2.148 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/9.2.148 )
```

Filtering Modules

- You can filter modules during test, this can be set by passing name argument to ModuleLoadTest. The name field takes a list of module names and Lmodule will test all module entries (all versions) found by spider
- In this example, we can test all cuda modules from a module tree as follows

```
>>> a = ModuleLoadTest("/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core",name=["cuda"])
Testing the Following Module Trees: /sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core
----------------------------------------------------------------------------------------------
PASSED -  Module Name: cuda/9.1.85 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/9.1.85 )
PASSED -  Module Name: cuda/11.0.1 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/11.0.1 )
PASSED -  Module Name: cuda/10.1.105 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/10.1.105 )
PASSED -  Module Name: cuda/11.0.2 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/11.0.2 )
PASSED -  Module Name: cuda/10.1.243 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/10.1.243 )
PASSED -  Module Name: cuda/9.2.148 ( modulefile=/sw/ascent/modulefiles/20180914/site/linux-rhel7-ppc64le/Core/cuda/9.2.148 )
```

Lmodule for environment modules

- Environment modules added support for parsing modules in json format in v4.5 see https://modules.readthedocs.io/en/latest/MIGRATING.html#json-format-output.
- This was inspired by issue #303 in order to test modules for entire stack
- You can automate module testing for environment-modules using Module class this required some bit of work.
- For full source see sourcefile

```
$ module avail --json bar | python -mjson.tool
{
    "/path/to/modulefiles": {
        "bar/2.3": {
            "name": "bar/2.3",
            "pathname": "/path/to/modulefiles/bar/2.3",
            "symbols": [
                "default"
            ],
            "type": "modulefile"
        },
        "bar/3.4": {
            "name": "bar/3.4",
            "pathname": "/path/to/modulefiles/bar/3.4",
            "symbols": [],
            "type": "modulefile"
        }
    }
}
```

```
import os
import re
import subprocess
import sys
from lmod.module import Module

modules = subprocess.getoutput("module av -t")
modules = modules.split()

pass_counter = 0
fail_counter = 0
total = 0

for module in modules:
    # output of module tree is as follows '/path/to/tree:' so we remove trailing colon
    tree = module[:-1]
    # skip entry when it's module tree
    if os.path.exists(tree):
        print(f"Skipping tree: {tree}")
        continue
    if re.search("(\(default\))$",module):
        module = module.replace('(default)','')

    cmd = Module(module,debug=True)
    ret = cmd.test_modules(login=True)
    total += 1
    # if returncode is 0 mark as PASS
    if ret == 0:
        pass_counter+=1
    else:
        fail_counter+=1

pass_rate = pass_counter * 100 / total
fail_rate = fail_counter * 100 / total

print ("-------- SUMMARY ---------")
print (f"Total Pass: {pass_counter}/{total}")
print (f"Total Failure: {fail_counter}/{total}")
print (f"PASS RATE: {pass_rate:.3f}")
print (f"FAIL RATE: {fail_rate:.3f}")
```

Lmodule Test

- We leverage Lmodule to automate module testing with buildtest
- In this example, we declare one test per module tree at Cori.
- We check output to see if we have a 100% pass rate
- In this test, we notice test **moduletest_craype_modulefiles** fails because module **perftools-lite/7.0.6** failed because perftools-base must be loaded

```
[DEBUG] Executing module command: bash -l -c "module purge && module load perftools-base/20.03.0  "
[DEBUG] Return Code: 0
[DEBUG] Executing module command: bash -l -c "module purge && module load perftools-base/20.06.0  "
[DEBUG] Return Code: 0
[DEBUG] Executing module command: bash -l -c "module purge && module load perftools-lite/7.0.6  "
[DEBUG] Return Code: 1
[DEBUG] Executing module command: bash -l -c "module purge && module load pmi/5.0.11  "
[DEBUG] Return Code: 0
```

```
-------- SUMMARY ----------
Total Pass: 139/145
Total Failure: 6/145
PASS RATE: 95.862
FAIL RATE: 4.138
```

```
siddiq90@cori09:~/buildtest-cori/modules> bash -l -c "module purge && module load perftools-lite/7.0.6"
------------------------------------------------------------------------------
Error: The Perftools module is available only after the perftools-base module
is loaded.

The Perftools-base module:
    - Provides access to Perftools man pages, Reveal and Cray Apprentice2
    - Does not alter compiling or program behavior
    - Makes the following instrumentation modules available:

perftools               - full support, including pat_build and pat_report
perftools-lite          - default CrayPat-lite profile
perftools-lite-events   - CrayPat-lite event profile
perftools-lite-gpu      - CrayPat-lite gpu kernel and data movement
perftools-lite-loops    - CrayPat-lite loop estimates (for Reveal)
perftools-lite-hbm      - CrayPat-lite memory bandwidth estimates (for Reveal)
perftools-preload       - CrayPat run-time support using pat_run utility
------------------------------------------------------------------------------
siddiq90@cori09:~/buildtest-cori/modules> echo $?
1
```

```
+----------------------+
| Stage: Running Test  |
+----------------------+

| Name                          | Executor   | Status   |   Returncode |
|-------------------------------|------------|----------|--------------|
| moduletest_opt_modulefiles    | local.bash | PASS     |            0 |
| moduletest_cray_modulefiles   | local.bash | PASS     |            0 |
| moduletest_cray_ari_modulefiles | local.bash | PASS   |            0 |
| moduletest_craype_modulefiles | local.bash | FAIL     |            0 |
| moduletest_ftg_modulefiles    | local.bash | PASS     |            0 |

+----------------------+
| Stage: Test Summary  |
+----------------------+

Executed 5 tests
Passed Tests: 4/5 Percentage: 80.000%
Failed Tests: 1/5 Percentage: 20.000%
```

```
version: "1.0"
buildspecs:
  moduletest_opt_modulefiles:
    description: Run module load test for tree /opt/modulefiles
    type: script
    executor: local.bash
    tags: [modules]
    run: MODULEPATH=/opt/modulefiles python moduletest.py
    status:
      regex:
        stream: stdout
        exp: "PASS RATE: 100.000"

  moduletest_cray_modulefiles:
    description: Run module load test for tree /opt/cray/modulefiles
    type: script
    executor: local.bash
    tags: [modules]
    run: MODULEPATH=/opt/cray/modulefiles python moduletest.py
    status:
      regex:
        stream: stdout
        exp: "PASS RATE: 100.000"


  moduletest_cray_ari_modulefiles:
    description: Run module load test for tree /opt/cray/ari/modulefiles
    type: script
    executor: local.bash
    tags: [modules]
    run: MODULEPATH=/opt/cray/ari/modulefiles python moduletest.py
    status:
      regex:
        stream: stdout
        exp: "PASS RATE: 100.000"


  moduletest_craype_modulefiles:
    description: Run module load test for tree /opt/cray/pe/modulefiles
    type: script
    executor: local.bash
    tags: [modules]
    run: MODULEPATH=/opt/cray/pe/modulefiles python moduletest.py
    status:
      regex:
        stream: stdout
        exp: "PASS RATE: 100.000"

  moduletest_ftg_modulefiles:
    description: Run module load test for tree /usr/common/ftg/modulefiles
    type: script
    executor: local.bash
    tags: [modules]
    run: MODULEPATH=/usr/common/ftg/modulefiles python moduletest.py
    status:
      regex:
        stream: stdout
        exp: "PASS RATE: 100.000"
```

Known Issue – Invalid Module Names

- In environment-modules version 3.x loading an unknown module will return a 0 exit code. This behavior is not present in environment-modules 4.x and Lmod

```
[shahzebsiddiqui@login1.ascent ~]$ module --version

Modules based on Lua: Version 8.2.10  2019-12-11 14:21 -06:00
    by Robert McLay mclay@tacc.utexas.edu

[shahzebsiddiqui@login1.ascent ~]$ module load gcc && module load X

Lmod is automatically replacing "xl/16.1.1-7" with "gcc/6.4.0".


Due to MODULEPATH changes, the following have been reloaded:
  1) spectrum-mpi/10.3.1.2-20200121

Lmod has detected the following error:  The following module(s) are unknown: "X"

Please check the spelling or version number. Also try "module spider ..."
It is also possible your cache file is out-of-date; it may help to try:
  $ module --ignore-cache load "X"

Also make sure that all modulefiles written in TCL start with the string #%Module


[shahzebsiddiqui@login1.ascent ~]$ echo $?
1
```

```
[shahzebsiddiqui@jlselogin2 ~]$ module --version
Modules Release 4.5.0 (2020-04-07)
[shahzebsiddiqui@jlselogin2 ~]$ module load gcc && module load X
ERROR: Unable to locate a modulefile for 'X'
[shahzebsiddiqui@jlselogin2 ~]$ echo $?
1
```

```
siddiq90@cori08:~> module -V
VERSION=3.2.11.4
DATE=2019-10-23

AUTOLOADPATH=undef
BASEPREFIX="/opt/cray/pe/modules"
BEGINENV=99
CACHE_AVAIL=undef
DEF_COLLATE_BY_NUMBER=undef
DOT_EXT=""
EVAL_ALIAS=1
HAS_BOURNE_FUNCS=1
HAS_BOURNE_ALIAS=1
HAS_TCLXLIBS=undef
HAS_X11LIBS=undef
LMSPLIT_SIZE=99999
MODULEPATH="/opt/cray/pe/modulefiles"
MODULES_INIT_DIR="/opt/cray/pe/modules/3.2.11.4/init"
PREFIX="/opt/cray/pe/modules/3.2.11.4"
TCL_VERSION="8.6"
TCL_PATCH_LEVEL="8.6.7"
TMP_DIR="/tmp"
USE_FREE=undef
VERSION_MAGIC=1
VERSIONPATH="/opt/cray/pe/modules/3.2.11.4"
WANTS_VERSIONING=1
WITH_DEBUG_INFO=undef

siddiq90@cori08:~> module load gcc && module load X
ModuleCmd_Load.c(244):ERROR:105: Unable to locate a modulefile for 'X'
siddiq90@cori08:~> echo $?
0
```

Conclusion

- Modules are the primary interface between users software stack, therefore it's important HPC sites test their software stack through module load testing.
- The first release v0.1.0 was available on Mar 25, 2020
- For further assistance join post your issue at https://github.com/buildtesters/lmodule/issues or join slack at **#lmodule** workspace
- References:
  - Docs: https://lmodule.readthedocs.io/
  - Examples: https://lmodule.readthedocs.io/en/latest/examples.html
  - GitHub: https://github.com/buildtesters/lmodule
  - PyPI: https://pypi.org/project/lmodule/
  - Slack: http://hpcbuildtest.slack.com/