DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

Modules v4

Yes, Environment Modules project is not dead

Xavier Delaruelle <xavier.delaruelle@cea.fr>

3rd EasyBuild User Meeting
January 30th 2018, SURFsara, Amsterdam

whoami

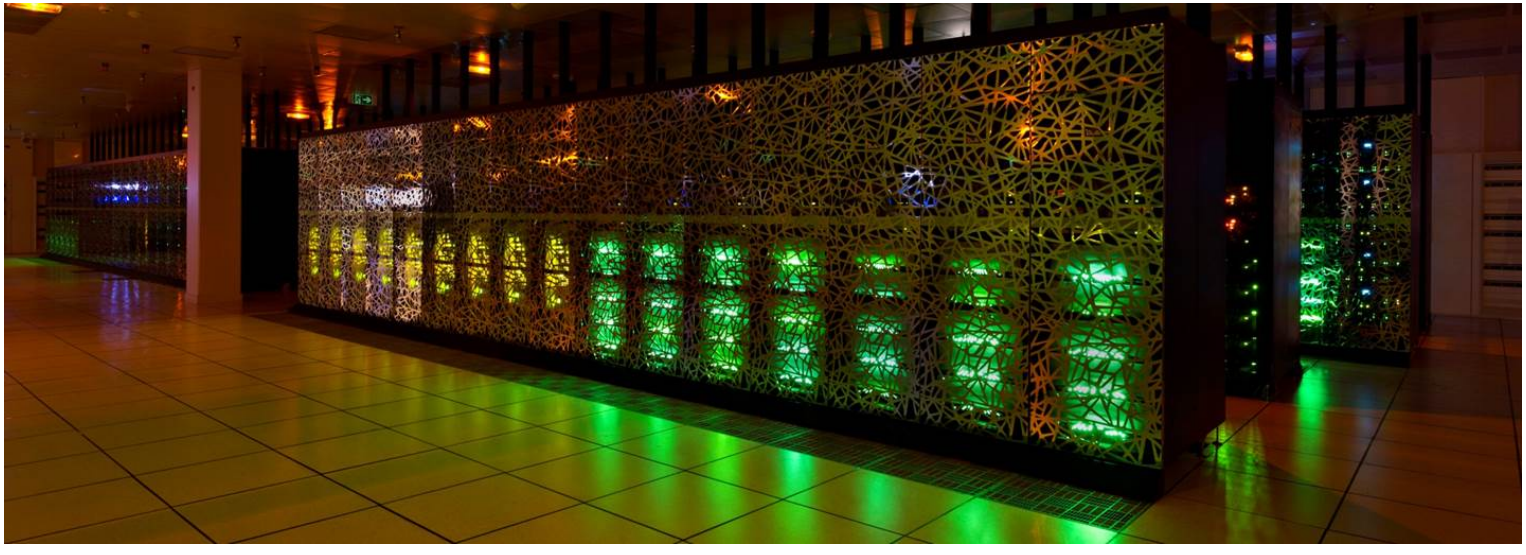
- I am Xavier Delaruelle
- Joined CEA in 2007 as HPC system administrator
- Operations manager of CEA's TGCC computing center, since 2016
- Involved in PRACE project (`http://www.prace-ri.eu/`) to federate user environment across the pan-European supercomputing infrastructure
- Got the key of the Environment Modules project from R.K. Owen in July 2017

CEA in brief

- French Alternative Energies and Atomic Energy Commission
- Key player in research, development and innovation in four main areas:
  - defence and security,
  - nuclear and renewable energies,
  - technological research for industry,
  - fundamental research in the physical sciences and life sciences.
- 9 research centres spread throughout France
- 16 010 technicians, engineers, researchers and staff as of 2016

High Performance Computing at CEA

- Significant investment in terms of research and development.
- Operates 2 large computing facilities
  - TERA: computing center for defence-related programmes
  - TGCC: designed to serve European academic research community and French industry



The CURIE supercomputer installed at TGCC

Environment Modules project

- 1991: Concept and initial implementation of the `module` command laid down
- 1999: Modules ported to Linux, version 3.0 written in C
- 2002: Introduction of `modulecmd.tcl`, pure-Tcl implementation
- 2012: Publication of Modules 3.2.10, last C-version feature release
- 2017: Modules 4.0.0 released, `modulecmd.tcl` now acts as main module command

Environment Modules project

- 1991: Concept and initial implementation of the `module` command laid down
- 1999: Modules ported to Linux, version 3.0 written in C
- 2002: Introduction of `modulecmd.tcl`, pure-Tcl implementation
- 2012: Publication of Modules 3.2.10, last C-version feature release
- 2017: Modules 4.0.0 released, `modulecmd.tcl` now acts as main module command

Why revive this project?

- Motivations
  - It is fun to hack on it
  - A will to open the solution to broader use-cases
    - There are many ways to setup a user environment
    - `module` should support that
- A sysadmin vision
  - `module` could leverage standard system tool approaches
  - To handle dependencies between modulefiles, why not proposing same behaviors than package manager tools (like `rpm`/`dnf` for instance)

Why revive this project?

- Motivations
  - It is fun to hack on it
  - A will to open the solution to broader use-cases
    - There are many ways to setup a user environment
    - `module` should support that
- A sysadmin vision
  - `module` could leverage standard system tool approaches
  - To handle dependencies between modulefiles, why not proposing same behaviors than package manager tools (like `rpm`/`dnf` for instance)

Why revive this project?

- Motivations
  - It is fun to hack on it
  - A will to open the solution to broader use-cases
    - There are many ways to setup a user environment
    - `module` should support that
- A sysadmin vision
  - `module` could leverage standard system tool approaches
  - To handle dependencies between modulefiles, why not proposing same behaviors than package manager tools (like `rpm`/`dnf` for instance)

But why `modulecmd.tcl`?

- Maintainability
  - Everything in 1 script file
  - Easy to fix and improve
  - I am not so proficient in C language
- Performances
  - No real performance loss, moving from a compiled binary to a script
  - Need to interpret Tcl script in any cases (for modulefiles)
  - A Tcl script that runs Tcl scripts: optimum conditions

But why `modulecmd.tcl`?

- Maintainability
  - Everything in 1 script file
  - Easy to fix and improve
  - I am not so proficient in C language
- Performances
  - No real performance loss, moving from a compiled binary to a script
  - Need to interpret Tcl script in any cases (for modulefiles)
  - A Tcl script that runs Tcl scripts: optimum conditions

But why `modulecmd.tcl`?

- Maintainability
  - Everything in 1 script file
  - Easy to fix and improve
  - I am not so proficient in C language
- Performances
  - No real performance loss, moving from a compiled binary to a script
  - Need to interpret Tcl script in any cases (for modulefiles)
  - A Tcl script that runs Tcl scripts: optimum conditions

The journey to Modules 4.0

- Test, fix and optimize
  - Close significant number of issues of C-version
  - Extend non-regression test suite (**from 250 to >5k tests**)
  - Almost complete refactoring of `modulecmd.tcl`
- Close existing gap with Modules 3.2
  - Implement missing *big* features of C-version
  - Align with most of the behaviors
  - Document remaining differences: `https://modules.readthedocs.io/en/stable/diff_v3_v4.html`

The journey to Modules 4.0

- Test, fix and optimize
  - Close significant number of issues of C-version
  - Extend non-regression test suite (**from 250 to >5k tests**)
  - Almost complete refactoring of `modulecmd.tcl`
- Close existing gap with Modules 3.2
  - Implement missing *big* features of C-version
  - Align with most of the behaviors
  - Document remaining differences: `https://modules.readthedocs.io/en/stable/diff_v3_v4.html`

DE LA RECHERCHE À L'INDUSTRIE
cea

Changes made to the project

- Development moved to GitHub (`https://github.com/cea-hpc/modules`)
- Add continious integration (Travis) 

- Monitor code coverage (Codecov) 

- Build and publish documentation (ReadTheDocs) 

- Semantic versioning approach to specify version numbers

v3.2 > v4.0: New features

- Additional shells supported (fish, lisp, tcl and R)
- Non-zero exit code in case of error
- Output redirect
- Filtering avail output
- Extended support for module alias and symbolic version
- Hiding modulefiles
- Improved modulefiles location
- Module collection
- Path variable element counter
- Optimized I/O operations
- Sourcing modulefiles

v3.2 > v4.0: New features

- Additional shells supported (fish, lisp, tcl and R)
- Non-zero exit code in case of error
- Output redirect
- Filtering avail output
- Extended support for module alias and symbolic version
- Hiding modulefiles
- Improved modulefiles location
- Module collection
- Path variable element counter
- Optimized I/O operations
- Sourcing modulefiles

Path variable element counter

- Count number of times a path entry is added to a path-like environment variable

```
$ echo $PATH
/bin:/usr/bin
$ module load t1 t2
$ echo $PATH
/bin:/usr/bin:/apps/common/bin
$ echo $PATH_modshare
/bin:1:/usr/bin:1:/apps/common/bin:2
```

- Keep path element when unloading if counter is greater than 1

```
$ module unload t1
$ echo $PATH
/bin:/usr/bin:/apps/common/bin
$ echo $PATH_modshare
/bin:1:/usr/bin:1:/apps/common/bin:1
```

- Remove path entry element if counter is equal to 1

```
$ module unload t2
$ echo $PATH
/bin:/usr/bin
```

Path variable element counter

- Count number of times a path entry is added to a path-like environment variable

```
$ echo $PATH
/bin:/usr/bin
$ module load t1 t2
$ echo $PATH
/bin:/usr/bin:/apps/common/bin
$ echo $PATH_modshare
/bin:1:/usr/bin:1:/apps/common/bin:2
```

- Keep path element when unloading if counter is greater than 1

```
$ module unload t1
$ echo $PATH
/bin:/usr/bin:/apps/common/bin
$ echo $PATH_modshare
/bin:1:/usr/bin:1:/apps/common/bin:1
```

- Remove path entry element if counter is equal to 1

```
$ module unload t2
$ echo $PATH
/bin:/usr/bin
```

Path variable element counter

- Count number of times a path entry is added to a path-like environment variable

```
$ echo $PATH
/bin:/usr/bin
$ module load t1 t2
$ echo $PATH
/bin:/usr/bin:/apps/common/bin
$ echo $PATH_modshare
/bin:1:/usr/bin:1:/apps/common/bin:2
```

- Keep path element when unloading if counter is greater than 1

```
$ module unload t1
$ echo $PATH
/bin:/usr/bin:/apps/common/bin
$ echo $PATH_modshare
/bin:1:/usr/bin:1:/apps/common/bin:1
```

- Remove path entry element if counter is equal to 1

```
$ module unload t2
$ echo $PATH
/bin:/usr/bin
```

Optimized I/O operations

- Improved path walk code to reduce the number of I/O operations when looking for modulefiles
- Total number of I/O calls divided by 2

	v3.2.10	**v4.0.0**	diff
access	2	193	+191
close	**799**	**338**	**-461**
fcntl	1	228	+227
fstat	232	108	-124
getdents	348	190	-158
ioctl	2	228	+226
lstat	6	677	+671
open	**797**	**332**	**-465**
read	**668**	**319**	**-349**
stat	**1985**	**416**	**-1569**
write	2125	4	-2121

# of I/O calls for a `module avail` on 200 modulefiles

Optimized I/O operations (2)

- Re-use when possible same Tcl interpreter to evaluate modulefiles
- Sanitize interpreter between each modulefile execution
- Total number of I/O calls divided by 3

	v3.2.10	**v4.0.0**	diff
access	299	209	-90
brk	712	52	-660
close	**1522**	**546**	**-976**
fcntl	704	427	-277
fstat	611	117	-494
getuid	298	3	-295
ioctl	707	431	-276
lstat	3333	1305	-2028
open	**1924**	**544**	**-1380**
read	**4634**	**729**	**-3905**
stat	**896**	**477**	**-419**
uname	298	3	-295
write	208	4	-204

# of I/O calls for a `module whatis` on 200 modulefiles

Sourcing modulefiles

- Source a modulefile rather loading it

```
$ module show /apps/initscript
-------------------
/apps/initscript:

append-path     PATH /apps/common/bin
-------------------
$ module source /apps/initscript
```

- Interpreted the same way but then not marked loaded

```
$ echo $PATH
/bin:/usr/bin:/apps/common/bin
$ echo $PATH_modshare
/bin:1:/usr/bin:1:/apps/common/bin:1
$ module list
No Modulefiles Currently Loaded.
```

Sourcing modulefiles (2)

- Open the path to new environment change ways in the future like sourcing modulefile when changing directory à la `direnv.net`
- Source a `.dirrc` modulefile when `cd` to the directory containing it

```
$ module list
No Modulefiles Currently Loaded.
$ cat $HOME/myworkdir/.dirrc
#%Module
module load liba/1.0 app/1.2
$ cd $HOME/myworkdir
$ module list
Currently Loaded Modulefiles:
 1) liba/1.0  2) app/1.2
$ cd -
$ module list
No Modulefiles Currently Loaded.
```

v4.0 > v4.1: New features

- Virtual modules
- Extend module command with site-specific Tcl code
- Quarantine mechanism to protect module execution
- Pager support
- Module function to return value in scripting languages
- New modulefile commands (`is-saved`, `is-used`, `is-avail`, `module-info loaded`)
- New module sub-commands (`append-path`, `prepend-path`, `remove-path`, `is-loaded`, `info-loaded`)
- Use variable reference in `MODULEPATH`

v4.0 > v4.1: New features

- Virtual modules
- Extend module command with site-specific Tcl code
- Quarantine mechanism to protect module execution
- Pager support
- Module function to return value in scripting languages
- New modulefile commands (`is-saved`, `is-used`, `is-avail`, `module-info loaded`)
- New module sub-commands (`append-path`, `prepend-path`, `remove-path`, `is-loaded`, `info-loaded`)
- Use variable reference in `MODULEPATH`

Virtual modules

- `module-virtual` associates a module name to a modulefile

```
$ cat /etc/modfiles/libraries/liba/.modulerc
#%Module1.0
module-virtual /1.0 .common
module-virtual /2.0 .common
$ cat /etc/modfiles/libraries/liba/.common
#%Module1.0
setenv TEST [module-info name]
```

- Appears or can be found with its virtual name.

```
$ module avail liba
------------ /etc/modfiles/libraries ------------
liba/1.0  liba/2.0
$ module load liba/1.0
$ module list
Currently Loaded Modulefiles:
 1) liba/1.0
```

- The target modulefile is the script interpreted

```
$ echo $TEST
liba/1.0
```

Virtual modules

- `module-virtual` associates a module name to a modulefile

```
$ cat /etc/modfiles/libraries/liba/.modulerc
#%Module1.0
module-virtual /1.0 .common
module-virtual /2.0 .common
$ cat /etc/modfiles/libraries/liba/.common
#%Module1.0
setenv TEST [module-info name]
```

- Appears or can be found with its virtual name.

```
$ module avail liba
------------- /etc/modfiles/libraries -------------
liba/1.0  liba/2.0
$ module load liba/1.0
$ module list
Currently Loaded Modulefiles:
 1) liba/1.0
```

- The target modulefile is the script interpreted

```
$ echo $TEST
liba/1.0
```

Virtual modules

- `module-virtual` associates a module name to a modulefile

```
$ cat /etc/modfiles/libraries/liba/.modulerc
#%Module1.0
module-virtual /1.0 .common
module-virtual /2.0 .common
$ cat /etc/modfiles/libraries/liba/.common
#%Module1.0
setenv TEST [module-info name]
```

- Appears or can be found with its virtual name.

```
$ module avail liba
------------- /etc/modfiles/libraries -------------
liba/1.0  liba/2.0
$ module load liba/1.0
$ module list
Currently Loaded Modulefiles:
 1) liba/1.0
```

- The target modulefile is the script interpreted

```
$ echo $TEST
liba/1.0
```

Virtual modules (2)

- Dynamically define available modulefiles depending on the situation
- A new perspective:
  - all modulefiles may only be defined in a central registry
  - queried by `MODULERCFILE` to get availabilities
  - no more modulepath to walk through

Extend module command with site-specific Tcl code

- `modulecmd.tcl` sources a `siteconfig.tcl` script at the beginning of its main procedure code

```
$ module -D -V
DEBUG CALLING /apps/Modules/libexec/modulecmd.tcl bash -D -V
DEBUG Source site configuration (/apps/Modules/etc/siteconfig.tcl)
...
```

- Enables to supersede any global variable or procedure definitions with site-specific code.

```
$ module load t1
load t1/1.0
```

- Will evolve to a more sophisticated hook system in a future release

Extend module command with site-specific Tcl code

- `modulecmd.tcl` sources a `siteconfig.tcl` script at the beginning of its main procedure code

```
$ module -D -V
DEBUG CALLING /apps/Modules/libexec/modulecmd.tcl bash -D -V
DEBUG Source site configuration (/apps/Modules/etc/siteconfig.tcl)
...
```

- Enables to supersede any global variable or procedure definitions with site-specific code.

```
$ module load t1
load t1/1.0
```

- Will evolve to a more sophisticated hook system in a future release

Extend module command with site-specific Tcl code

- `modulecmd.tcl` sources a `siteconfig.tcl` script at the beginning of its main procedure code

```
$ module -D -V
DEBUG CALLING /apps/Modules/libexec/modulecmd.tcl bash -D -V
DEBUG Source site configuration (/apps/Modules/etc/siteconfig.tcl)
...
```

- Enables to supersede any global variable or procedure definitions with site-specific code.

```
$ module load t1
load t1/1.0
```

- Will evolve to a more sophisticated hook system in a future release

Roadmap

- A feature release cut every 4 months
- Bug fix releases in-between if necessary

Next releases

2018-05 v4.2.0

2018-09 v4.3.0 (or v5.0.0)

2019-01 v5.0.0 (or v5.1.0)

Expectation for v4.2

- Meta alias or Package
- Loaded module requirement awareness
- Improved `conflict` and `prereq` specifications
- Resolved alias/symbolic version on `conflict` and `prereq`
- Automatic dependency management **technology preview**
- By-pass `conflict` constraint with `–force`
- By-pass `prereq` constraint with `–nodeps`

Expectation for v4.2

- Meta alias or Package
- Loaded module requirement awareness
- Improved `conflict` and `prereq` specifications
- Resolved alias/symbolic version on `conflict` and `prereq`
- Automatic dependency management **technology preview**
- By-pass `conflict` constraint with `–force`
- By-pass `prereq` constraint with `–nodeps`

Loaded module requirement awareness

- Once loaded modules loose track of their `conflict` and `prereq` requirements

```
$ module show libb
_____________________

/etc/modfiles/libraries/libb/.common:

conflict        liba
_____________________

$ module load libb liba
$ module list
Currently Loaded Modulefiles:
 1) libb/2.0   2) liba/2.0
```

- Loading requirement will help keeping track of it (_LMCONFLICT_ and _LMPREREQ_ in addition to _LMFILES_)

```
$ module load libb liba
ERROR: WARNING: liba/2.0 cannot be loaded due to a conflict.
HINT: Might try "module unload libb/2.0" first.
$ module list
Currently Loaded Modulefiles:
 1) libb/2.0
```

Loaded module requirement awareness

- Once loaded modules loose track of their `conflict` and `prereq` requirements

```
$ module show libb
_____________________

/etc/modfiles/libraries/libb/.common:

conflict        liba
_____________________

$ module load libb liba
$ module list
Currently Loaded Modulefiles:
 1) libb/2.0   2) liba/2.0
```

- Loading requirement will help keeping track of it (_LMCONFLICT_ and _LMPREREQ_ in addition to _LMFILES_)

```
$ module load libb liba
ERROR: WARNING: liba/2.0 cannot be loaded due to a conflict.
HINT: Might try ”module unload libb/2.0” first.
$ module list
Currently Loaded Modulefiles:
 1) libb/2.0
```

Automatic dependency management

- Please load it yourself!

```
$ module load app
WARNING: app/2.0 cannot be loaded due to missing prereq.
HINT: the following module must be loaded first: libb
```

- Should be handled without manual intervention

```
$ module load app
load libb/2.0
load app/2.0
$ module list
Currently Loaded Modulefiles:
 1) libb/2.0   2) app/2.0
```

Automatic dependency management

- Please load it yourself!

```
$ module load app
WARNING: app/2.0 cannot be loaded due to missing prereq.
HINT: the following module must be loaded first: libb
```

- Should be handled without manual intervention

```
$ module load app
load libb/2.0
load app/2.0
$ module list
Currently Loaded Modulefiles:
 1) libb/2.0   2) app/2.0
```

Automatic dependency management (2)

- Advanced behaviors
  - Changing requirement reloads dependent modules
  - Unload dependent module unloads automatically loaded dependencies
  - Save in collection only what was asked by the user
- Handle multiple dependency chains as long as their is no conflict between them
- This concept is in production at our site since 2014

Conclusion

- A lot of things done since 1 year
- Many ideas in stock
- So this is just the beginning
- Stay tuned

Thanks for your attention

- Website: `http://modules.sourceforge.net/`
- Code: `https://github.com/cea-hpc/modules`
- Documentation: `https://modules.readthedocs.io`
- Questions, feedback, new use-cases, want to participate: `modules-interest@lists.sourceforge.net`

Commissariat à l’énergie atomique et aux énergies alternatives
Centre de Bruyères-le-Châtel │ 91297 Arpajon Cedex
T. +33 (0)1 69 26 40 00 │ F. +33 (0)1 69 26 40 00
Établissement public à caractère industriel et commercial
RCS Paris B 775 685 019

DAM
DIF
DSSI