Les Modules PlaFRIM

Nathalie Furmento et Brice Goglin

14 février 2020

Contents

1 Introduction

- Dynamic modification of a user's environment via modulefiles.
- Each modulefile contains the information needed to use the specific library/compiler/... by altering or setting shell environment variables such as `PATH`, `MANPATH`, etc.
- Different versions of applications.
- Metamodules can be used to load an entire suite of different applications.

2 Users modules

- All users can install modules, one only needs to be added to the Unix group `plafrim-dev` (ticket to `plafrim-support@inria.fr`) ...
- ... and install modules in `/cm/shared/dev/modules`
- But there is some rules to follow ...

3 Architecture

- For modules which can run on all nodes, the directory `generic` should be used, with the sub-directories `apps` and `modulesfiles`.
- For modules aimed to specific architectures, the following directories should be used
  - `intel/haswell` for the nodes `miriel` and `sirocco[01-05]`
  - `intel/broadwell` for the nodes `sirocco[07-13]`
  - `intel/skylake` for the nodes `bora`
  - `intel/knightslanding` for the nodes `kona`
- When connecting to a node, the environnment variable `MODULEPATH` contains the directory `generic` and the node-specific directory `manufacturer/chip`

4 Categories

- In order to increase and to ease the use of the modules on the platform, modules are grouped within categories. Each module belongs to a specific category, which can be for example `trace` or `statistics`.
- The module naming policy is as follows:

  category/*module*/*option*/*version*
- the number of options being between 0 and as many as you want.

  `partitioning/scotch/int32/6.0.4`

  `partitioning/scotch/int64/6.0.4`

5 File System

- Modules files go in `/cm/shared/dev/modules/XXX/modulefiles` by following the architecture and the naming policies.
  Generic module `fxt` in the category `trace` with version `0.3.9` and without any specific option

  → `/cm/shared/dev/modules/generic/modulefiles/trace/fxt/0.3.9`

- Installation module files go in `/cm/shared/dev/modules/XXX/apps` with the same architecture and naming policies.
  → `/cm/shared/dev/modules/generic/apps/trace/fxt/0.3.9/`

- Make sure all the files can be read by anyone on the platform.

- Write a real description in whatis, the command `module show/whatis xxx` should say something more meaningful than `loads the xxx environment`

6 How to create a module

- Let's assume we install a `generic` module
- Install your software in the appropriate sub-directory in `/cm/shared/dev/modules/generic/apps`
- For example in the directory `/cm/shared/dev/modules/generic/apps/tools/boost/1.71.0/`
- The module file is `/cm/shared/dev/modules/generic/modulefiles/tools/boost/1.71.0`

```
proc ModulesHelp { } {
  puts stderr "\tAdds boost 1.71.0 to your environment variables,"
}

module-whatis "adds boost 1.71.0 debug tool to your environment variables"

set             version         1.71.0
set             prefix          /cm/shared/dev/modules/generic/apps/tools/boost/
set             root            $prefix/$version

#%Module

#path added in the beginning
prepend-path   CPATH  $root/include
prepend-path   LIBRARY_PATH  $root/lib
prepend-path   LD_LIBRARY_PATH  $root/lib
setenv   BOOST_ROOT  $root
```

- Set the correct permissions

```
$ module load tools/module_cat
$ module_perm my_directory
```

- If your module depends on other modules `/cm/shared/dev/modules/generic/modulefiles/perftools/simgrid/3.24`
  - specify the dependencies

    ```
    prereq compiler/gcc
    prereq tools/boost/1.71.0
    ```

  - and use their informations

    ```
    set     prefix          /cm/shared/dev/modules/generic/apps/perftools/simgrid/$version/install/gcc_$env(GCC_VER)
    ```

  - The module can have different versions
    - `/cm/shared/dev/modules/generic/apps/perftools/simgrid/3.24/install/gcc_8.2.0/`
    - `/cm/shared/dev/modules/generic/apps/perftools/simgrid/3.24/install/gcc_9.2.0/`
- To avoid loading 2 versions of the same module

  ```
  conflict runtime/starpu
  ```

- When a module depends on the versions of other modules

  ```
  set     prefix          /cm/shared/dev/modules/generic/apps/perftools/simgrid/$version/install/gcc_$env(GCC_VER)
  if {![file exists $prefix]} {
      puts stderr "\t[module-info name] Load Error: $prefix does not exist"
      break
      exit 1
  }
  ```

- What are the useful variables?
  - Path to development headers (for compiling)

```
CPATH
FPATH
INCLUDE
C_INCLUDE_PATH
CPLUS_INCLUDE_PATH
OBJC_INCLUDE_PATH
```

  - Path to libraries (for linking)

```
LIBRARY_PATH
```

  - Path to tools and libraries (for running)

```
PATH
LD_LIBRARY_PATH
```

  - pkg-config to simplifying build systems (point to directories with .pc files)

```
prepend-path    PKG_CONFIG_PATH     $prefix/lib/pkgconfig
```

  - Manpages

```
append-path     MANPATH         $man_path
append-path     MANPATH         $man_path/man1
append-path     MANPATH         $man_path/man3
append-path     MANPATH         $man_path/man7
```

  - Some modules define specific variables, likely because one random project ever decided to use that

```
setenv            CUDA_INSTALL_PATH   $root
setenv            CUDA_PATH           $root
```

```
setenv            CUDA_SDK            $root
prepend-path      CUDA_INC_PATH       $root/include

setenv          HWLOC_HOME          $prefix

setenv         MPI_HOME          $prefix
setenv         MPI_RUN           $prefix/bin/mpirun
setenv         MPI_NAME          $name
setenv         MPI_VER           $version
```

- Which version gets selected by default during load?
  - The default is in (reverse) alphabetical order

```
$ module av formal/sage

------------------ /cm/shared/dev/modules/generic/modulefiles ------------------
formal/sage/7.0 formal/sage/8.9 formal/sage/9.0
$ module av 2>&1 | grep sage
formal/sage/7.0
formal/sage/8.9
formal/sage/9.0
$ module load formal/sage
$ module list
Currently Loaded Modulefiles:
  1) formal/sage/9.0
```

  - Another default version may be enforced. Useful if your last beta release isn't stable or backward compatible but still needed for some hardcode users.

```
$ cat /cm/shared/dev/modules/generic/modulefiles/hardware/hwloc/.version
#%Module1.0#
set ModulesVersion "2.1.0"
```

7 More on environment dependent modules

```
$ module_grep starpu
runtime/starpu/1.3.2/mpi
runtime/starpu/1.3.2/mpi-fxt
runtime/starpu/1.3.3/mpi
runtime/starpu/1.3.3/mpi-cuda
runtime/starpu/1.3.3/mpi-cuda-fxt
runtime/starpu/1.3.3/mpi-fxt
```

- StarPU has different module files which differ in the `prereq` commands and the prefix setting.

```
> prereq compiler/cuda/10.1
> prereq trace/fxt/0.3.9
< set     prefix          /cm/shared/dev/modules/generic/apps/runtime/starpu/1.3.3/gcc@9.2.0-hwloc@2.1.0-openmpi@4.0.2
> set     prefix          /cm/shared/dev/modules/generic/apps/runtime/starpu/1.3.3/gcc@8.2.0-hwloc@2.1.0-openmpi@4.0.1-cuda@10.1
> set     prefix          /cm/shared/dev/modules/generic/apps/runtime/starpu/1.3.3/gcc@8.2.0-hwloc@2.1.0-openmpi@4.0.1-cuda@10.1-fxt@0.3.9
```

- Create a module file with all the cases

```
if {![ is-loaded compiler/gcc ]} {
   module load compiler/gcc
}
if {![ is-loaded hardware/hwloc ]} {
   module load hardware/hwloc/2.1.0
}

conflict runtime/starpu

set cuda ""
set fxt ""
if {[ is-loaded compiler/cuda/10.1 ]} {
  set cuda -cuda@10.1
}
if {[ is-loaded trace/fxt/0.3.9 ]} {
  set fxt -fxt@0.3.9
}

# for Tcl script use only
set     name            starpu
set     version         1.3.3

set     prefix          /cm/shared/dev/modules/generic/apps/runtime/$name/$version/gcc@$env(GCC_VER)-hwloc@2.1.0-openmpi@4.0.1${cuda}${fxt}
```

- Different environments lead to a different version of StarPU to be used:
  - 1st case

```
$ module purge
$ module load runtime/starpu/42
$ module list
Currently Loaded Modulefiles:
  1) compiler/gcc/9.2.0     2) hardware/hwloc/2.1.0   3) runtime/starpu/42
$ module show runtime/starpu/42
...
setenv  STARPU_DIR /cm/shared/dev/modules/generic/apps/runtime/starpu/1.3.3/gcc@9.2.0-hwloc@2.1.0-openmpi@4.0.1
```

  - 2nd case

```
$ module purge && module load compiler/gcc/8.2.0 compiler/cuda/10.1 runtime/starpu/42
$ module show runtime/starpu/42
...
setenv  STARPU_DIR /cm/shared/dev/modules/generic/apps/runtime/starpu/1.3.3/gcc@8.2.0-hwloc@2.1.0-openmpi@4.0.1-cuda@10.1
```

  - 3rd case

```
$ module purge && module load compiler/gcc/8.2.0 compiler/cuda/10.1 trace/fxt runtime/starpu/42
$ module show runtime/starpu/42
...
setenv  STARPU_DIR /cm/shared/dev/modules/generic/apps/runtime/starpu/1.3.3/gcc@8.2.0-hwloc@2.1.0-openmpi@4.0.1-cuda@10.1-fxt@0.3.9
```

8 More modules commands

- `https://modules.readthedocs.io/en/latest/modulefile.html`

9 tools/module_cat

- `module load tools/module_cat`
- You can then list all the modules whose name contains a specific string.

```
$ module_grep scotch
partitioning/scotch/int32/5.1.12
partitioning/scotch/int32/6.0.9
partitioning/scotch/int64/5.1.12
partitioning/scotch/int64/6.0.9
partitioning/scotch_with_esmumps/6.0.7
```

- List all the categories

```
$ module_list
------------------------------------------------------------------------------------------ /cm/shared/modules/generic/modulefiles ------------
build compiler language mpi tools
------------------------------------------------------------------------------------------ /cm/shared/modules/intel/haswell/modulefiles ------
compiler linalg
------------------------------------------------------------------------------------------ /cm/shared/dev/modules/generic/modulefiles --------
benchmark dnn gis partitioning tools
bigdata editor hardware perftools trace
build fft io physics visu
compiler formal language runtime
dataencoding genome linalg scm
```

- List the content of a given category

```
$ module_list hardware
------------------------------------------------------------------------------------------ /cm/shared/dev/modules/generic/modulefiles --------
hardware/hwloc hardware/libpciaccess
```

- Restrict MODULEPATH to a specific set of categories

```
$ module_restrict mpi compiler && echo $MODULEPATH
:/cm/shared/modules/generic/modulefiles/compiler:/cm/shared/modules/generic/modulefiles/mpi:/cm/shared/modules/intel/haswell/modulefiles/compiler
```

- Add some categories

```
$ module_add tools && echo $MODULEPATH
[module_add] adding /cm/shared/modules/generic/modulefiles/tools
[module_add] adding /cm/shared/dev/modules/generic/modulefiles/tools
/cm/shared/dev/modules/generic/modulefiles/tools:/cm/shared/modules/generic/modulefiles/tools:::/cm/shared/modules/generic/modulefiles/compiler:/
```

- Remove some categories

```
$ module_rm mpi compiler && echo $MODULEPATH
/cm/shared/dev/modules/generic/modulefiles/tools:/cm/shared/modules/generic/modulefiles/tools:
```

- Help for the different options are displayed by calling

```
$ module_help
module_<command> with <command> being:
      help                                 - display help
      list [<category>]                    - list categories (from the top level by default, or the given category)
      init                                 - list categories from the initial setup
      reset                                - reset initial MODULEPATH
      add <category1> ... <categoryn>      - add given category/ies to MODULEPATH
      rm <category1> ... <categoryn>       - remove given category/ies from MODULEPATH
      restrict <category1> ... <categoryn> - only keep given category/ies in MODULEPATH
```