Environment Modules

Introduction

Environment modules ("module" command) is an OPTIONAL tool in RCCS
You can set PATH or LD_LIBRARY_PATH environment variables in your setting files and
load config files provided by applications as it always has been.

Pros of environment modules in RCCS

- able to simplify complicated settings of certain applications.
- not need to care about the location of setting files of applications.
- easy to switch environment (e.g. CUDA 8.0/9.1 or Intel MPI/Open MPI).

Cons of environment modules in RCCS

- If your login shell OR your job script is csh, additional line is required in job script.
  - If both of login shell and job scripts are written in sh/bash, no additional lines required.
  - if your job script is csh script, "source /etc/profile.d/modules.csh" is required in job submission script before using module command
  - if your login shell is csh but the job script is bash script, you need to add "source /etc/profile.d/modules.sh" in the script.
- if you already have very complicated settings, it is terribly difficult to migrate to module.
- no significant advantage for certain applications (most versions of gromacs etc.).

Basic Usage

Loading/unloading modules can be done by "module" command. This command is available both in frontend and computation nodes.
If you are using csh script for job submission, you may need to pay special attention in your script.
Please take a look at the csh script example below.

- "load" module (some of modules are automatically loaded when you login)

```
% module load (module name)
```

- "unload" (or remove) modules. In some cases, module dependencies could broken by this command due to their complexity. If you met the situation, please "purge" first and then "load" modules.

```
% module unload (module name)
```

- "purge"; unload all the modules

```
% module purge
```

- show available modules (if you omit path, all the available modules will be shown)

```
% module avail (path; optional)
```

- show "list" of loaded modules

```
% module list
```

- show brief explanations of modules (length of about single line; if you omit module name, messages for all the modules will be shown)

```
% module whatis (module name; optional)
```

- somewhat detailed explanation of a module (sample files location might be shown for some modules)

```
% module help (module name)
```

module default module

The default version is specified for most modules. If you load a module without specifying module versions,
the default version will be loaded.
For example, in Intel MPI,

```
% module purge
% module avail mpi/intelmpi
---------- /local/apl/lx/modules/apl ----------
mpi/intelmpi/2017.3.196 mpi/intelmpi/2019
mpi/intelmpi/2018.2.199 mpi/intelmpi/5.0.2.044
% module load mpi/intelmpi
% module list
Currently Loaded Modulefiles:
  1) mpi/intelmpi/2018.2.199
```

For some modules, the default version might be shown in the output of "module avail" command.
Default version might be specified even when (default) is not shown in "module avail".
You can check the default version by checking .version file in module directories.
In the case above, default version for mpi/intempi is specified in /local/apl/lx/modules/apl/mpi/intelmpi/.version.
This file contains the following information:

```
% cat /local/apl/lx/modules/apl/mpi/intelmpi/.version
#%Module 1.0
```

```
set ModulesVersion "2018.2.199"
```

In case of unloading, this kind of omitted notation works. For example,

```
% module list
Currently Loaded Modulefiles:
  1) mpi/intelmpi/2018.2.199
% module unload mpi
% module list
No Modulefiles Currently Loaded.
```

Sample Usage in Job Submission Script

Here are the examples for amber18-bf1 (which requires intel parallelstudio and cuda 9.1).
These kinds of sample scripts can be found in samples/ directory of each application (file names are *-module.*sh).

Conventional Way (sh)

```
#!/bin/sh
#PBS select=ncpus=1:mpiprocs=1:ompthreads=1:jobtype=gpu:ngpus=1
#PBS -l walltime=72:00:00
if [ ! -z $PBS_O_WORKDIR ]; then
  cd $PBS_O_WORKDIR
fi
export PATH=/local/apl/lx/cuda-9.1/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/local/apl/lx/cuda-9.1/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
. /local/apl/lx/amber18-bf1/amber.sh
pmemd.cuda -O -i mdin .....
```

Using Environment Modules

```
#!/bin/sh
#PBS select=ncpus=1:mpiprocs=1:ompthreads=1:jobtype=gpu:ngpus=1
#PBS -l walltime=72:00:00
if [ ! -z $PBS_O_WORKDIR ]; then
  cd $PBS_O_WORKDIR
fi
module load amber/18/bugfix1
pmemd.cuda -O -i mdin .....
```

If your login shell is csh, you need to add "source /etc/profile.d/modules.sh"(source can be replaced with ".") before module load command.

Using Environment Modules in csh job script

For csh/tcsh, module command is defined as an alias command in /etc/profile.d/modules.csh.
Although you don't need a special setting in interactive shell, you should source /etc/profile.d/modules.csh
if you want to use module command in job submission script.

```
#!/bin/csh -f
#PBS select=ncpus=1:mpiprocs=1:ompthreads=1:jobtype=gpu:ngpus=1
#PBS -l walltime=72:00:00
if ( $?PBS_O_WORKDIR ) then
  cd $PBS_O_WORKDIR
endif
source /etc/profile.d/modules.csh # required!
module load amber/18/bugfix1
pmemd.cuda -O -i mdin .....
```

RCCS specifics

Available modules can be listed by "module avail" command.

Modules automatically loaded upon login

(on 2018/6/30)
The following modules are loaded when you logged in.

- intel_parallelstudio/2018update2 (Intel Parallel Sutdio 2018)
- pgilic (PGI license setting)
- pgi/18.1 (PGI compiler 18.1)
- allinea/7.1 (Allinea Forge 7.1)
- cuda/8.0 (NVIDIA CUDA 8.0)

Rule for module names

There are no strict rules in module name definition. Typically, module names are defined as
package name/version(-revision)/(revision or compiler type).

e.g.:
amber/18/bugfix1
gaussian/g16/b01

gromacs/2016.5/intel
vmd/1.9.3

Module Tree Overview

- suite

Including software suites.
Intel Parallel Studios and Software Collections are included in this category.
Note: Intel Parallel Studio modules may conflict with its individual component such as intel compiler.

- comp

Compiler/runtime/programming environment including NVIDIA CUDA.

- apl

Miscellaneous applications.
MPI environments (Intel MPI, Open MPI) belong to this category.

- apl_util

Utility applications.

- apl_ex

Applications which will be used in computation nodes.
Dependencies for modules in this category will be automatically solved when loading.
(Compilers and MPI environment will also be loaded without asking you.)

- lib

Libraries.

- misc

Miscellaneous tools. You may not need to manually load/unload modules in this category.
You might want to load "inteldev" module if you want to use advisor or vtune without loading whole intel parallel studio.

Reference

- Official Site