Environment Modules

はじめに

environment modules(moduleコマンド)の利用は必須ではありません。
これまで通り、手動でPATHやLD_LIBRARY_PATHを設定したり、アプリケーション側で提供している
設定ファイル(*.sh,*.csh)を読み込んだりして使うことができます。

moduleコマンドを使うメリット

- 複雑な初期設定を要するアプリケーションについての設定がすっきりする
- アプリケーション側の設定ファイルの場所を意識する必要がなくなる
- 依存関係を切り替える時(例: CUDA 8.0/9.1)の手間が減る、考える必要がなくなる

moduleコマンドのデメリットや注意点

- ログインシェルやジョブスクリプトがcshの場合に一手間多い
  - ログインシェルがbash系で、ジョブスクリプトもsh, bash, zshならば特別な設定は必要ありません
  - csh のジョブスクリプトでは module コマンド利用前に source /etc/profile.d/modules.csh を明示的に実行する必要があります
  - ログインシェルが csh で、ジョブスクリプトを /bin/sh にする場合にも、source /etc/profile.d/modules.sh (sourceは.で書いても可)を事前に実行する必要があります
- 既に複雑な環境を自身のホームにある設定ファイルで構築している場合は移行が困難
- 設定が簡略化されているアプリケーションについてはメリットが極めて薄い

基本的な利用方法

moduleコマンドで操作します。フロントエンドノード、演算ノードで共通に利用できます。
ただし、cshスクリプトでは特殊な注意が必要です。ページ下部をご覧ください。
ログイン時に自動でいくつかのパッケージが読み込まれていることにご注意ください。

- モジュールの読み込み(load)

```
% module load (モジュール名)
```

- モジュールのアンロード(unload; removeも可)。依存関係が複雑なものについては環境が壊れる可能性があります。そのような時は申し訳ありませんが一度purgeしてから必要なモジュールをloadするようにしてください。

```
% module unload (モジュール名)
```

- 全モジュールのアンロード

```
% module purge
```

- 利用可能なモジュールの表示(パスを指定しない場合は全て)

```
% module avail (パス; オプション)
```

- 現在読み込んでいるモジュールの表示

```
% module list
```

- モジュールの簡単な説明(一行程度; モジュール名を指定しない場合には全てについて表示します)

```
% module whatis (モジュール名; オプション)
```

- 比較的詳細なモジュールの説明(一部のモジュールではサンプルファイルの場所等も表示します)

```
% module help (モジュール名)
```

moduleのデフォルトバージョン

大抵のモジュールではデフォルトのバージョンが設定されています。バージョンを指定しないでloadすると、このデフォルトのバージョンが使われます。
インテルMPIの場合を例にしてみます。

```
% 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
```

デフォルトバージョンはmodule availした時に表示される場合もありますが、されない場合もあります。
表示されない場合でも、moduleの定義されているディレクトリにある.versionというファイルを見れば確認できます。
今回の場合、mpi/intelmpiに対するデフォルトの定義は、/local/apl/lx/modules/apl/mpi/intelmpi/.versionにあります。
ファイルの中身を見ると次のようになっています。

```
% cat /local/apl/lx/modules/apl/mpi/intelmpi/.version
#%Module 1.0
set ModulesVersion "2018.2.199"
```

この設定によって、mpi/intelmpiを指定した時には2018.2.199のバージョンが使われます。
なお、moduleをunloadする際にも簡略化した表記が有効です。

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

ジョブスクリプトでの利用例

amber18-bf1(インテル系ライブラリと非デフォルト環境のCUDA9.1を利用する場合)を利用する場合の比較を以下に示します。
このようなサンプルは各アプリケーションのsamples/ディレクトリにも置いてあります(*-module.*sh という名前です)。

これまでの方法(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 .....
```

shスクリプトでmodule利用

```
#!/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 .....
```

ログインシェルが csh で、スクリプトだけsh, bashの場合は、module コマンド実行前に source /etc/profile.d/modules.sh が必要です

cshジョブスクリプトでmodule利用

cshの場合、moduleコマンドは/etc/profile.d/modules.cshでaliasとして定義されています。
ログインして対話型シェルで使う場合には特に設定は要りませんが、
ジョブスクリプトで利用する場合には明示的に/etc/profile.d/modules.cshを読み込む必要があります。

```
#!/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での設定

利用可能なモジュールリストは module avail コマンドで参照できます。

ログイン時に自動的に読み込まれるモジュール

(2018/6/30現在)
初期状態で自動的に読み込まれるモジュールは以下のようになっています。
これまでデフォルトで設定されていたPATH等の設定に対応しています。

- intel_parallelstudio/2018update2 (インテル Parallel Sutdio 2018)
- pgilic (PGI ライセンス設定)
- pgi/18.1 (PGI コンパイラ 18.1)
- allinea/7.1 (Allinea Forge 7.1)
- cuda/8.0 (NVIDIA CUDA 8.0)

モジュールの命名規則

厳密な規則はありませんが、原則的には
パッケージ名/バージョン(-リビジョン)/(リビジョン or コンパイラ種類)
と言った順序で並べるようにしています。

例:

amber/18/bugfix1
gaussian/g16/b01
gromacs/2016.5/intel
vmd/1.9.3

構成と簡単な説明

- suite

統合的なモジュールです。
ソフトウェアコレクション(scl)や、インテルの Parallel Studio を一括で読み込むための module 等があります。
ここにある intel_parallelstudio は apl や comp、lib に含まれるコンパイラ等の個別パッケージと conflict する可能性があります。

- comp

CUDAも含めた各種コンパイラ、プログラム言語の環境です。

- apl

雑多なアプリケーションです。
MPI 環境(Intel MPI, Open MPI)もここにあります。

- apl_util

ユーティリティ系のアプリケーションです

- apl_ex

実際に計算に使うアプリケーションはここに配置しています。
ここに含まれるmodule群は、loadが失敗しないよう、ほぼ強制的に依存関係を解決します。

- lib

各種ライブラリです

- misc

その他の雑多な module です。明示的に load/unload する必要はほとんどないはずです。
parallelstudio全体を読み込まずに advisor、vtune 等を使いたい場合があれば、inteldev モジュールを load してください。

参考リンク

- 公式サイト(英語)