Ustawianie środowiska aplikacji - Modules

Podstawowe informacje

- Mechanizm *Modules* umożliwia łatwą i dynamiczną modyfikację zmiennych powłoki (ścieżki wyszukiwania aplikacji itp.) dostosowując ją do specyficznych wymagań danego pakietu oprogramowania.
- Dla każdego z pakietów oprogramowania dostępnego w Infrastrukturze PLGrid zdefiniowano odpowiadający mu moduł (lub moduły, jeśli istnieje więcej niż jedna wersja danego pakietu).

> Zalecamy wykorzystanie Katalogu Aplikacji i Usług do wyszukiwania dostępnego w PLGrid oprogramowania oraz modułów.

Komenda module

Dla komendy `module` należy podać parametr określający akcję. Najczęściej wykorzystywane opcje to:

- `module add <modu>` lub `module load <modu>` – załadowanie modułu danego programu
- `module rm <modu>` lub `module unload <modu>` – usunięcie modułu danego programu
- `module list` – wyświetlenie listy aktualnie załadowanych modułów
- `module avail` – wyświetla listę wszystkich dostępnych modułów
- `module avail <nazwa>` – wyświetla listę wszystkich dostępnych wersji oprogramowania o nazwie rozpoczynającej się od `<nazwa>`
- `module purge` – usunięcie wszystkich załadowanych modułów
- `module show <modu>` lub `module display <modu>` – wyświetl informacje nt. danego modułu
- `module switch <modu-1> <modu-2>` – wymiana modułów w powłoce

Schemat nazw modułów

Nazwy modułów dla aplikacji naukowych budowane są według schematu `plgrid/apps/nazwa-programu/wersja`.

W przypadku bibliotek schemat ma postać `plgrid/libs/nazwa-biblioteki/wersja`, a dla programów narzędziowych (np. język oprogramowania Python) `plgrid/tools/nazwa-narzedzia/wersja`.

Zarówno dla pakietów oprogramowania jak i bibliotek przygotowano wersję też domyślną i przy jej ładowaniu ostatni człon modułu (tj. `wersj`) można pominąć.

Przykładowe użycie

- Załadowanie domyślnej wersji kompilatora Intel

```
module add plgrid/tools/intel
```

- Załadowanie kompilatora Intel w wersji 13.0

```
module add plgrid/tools/intel/13.0
```

- Spis wszystkich dostępnych wersji kompilatorów Intel

```
module avail plgrid/tools/intel
```

- Zamiana wersji Matlab z wersji R2012b na R2013b

```
module switch plgrid/apps/matlab/R2012b plgrid/apps/matlab/R2013b
```

- Usunięcie ścieżek do kompilatora Intel w wersji 13.0 z środowiska

```
module rm plgrid/tools/intel/13.0
```

- Przykładowy skrypt SLURM wykorzystujący polecenie Module do załadowania programu Matlab w domyślnej wersji i wykonania obliczeń zawartych w pliku matlab.m (skrypt to matlab.pbs)

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --ntasks-per-node=2
#SBATCH -t 10:00
#SBATCH -p plgrid-testing

#przejdz do katalogu na pliki tymczasowe zadania
cd $TMPDIR

#ustaw srodowisko uruchomieniowe dla Matlab w wersji domyslnej
module add plgrid/apps/matlab

#uruchom program
matlab < $SLURM_SUBMIT_DIR/matlab.m > $SLURM_SUBMIT_DIR/matlab.out

#skasuj katalog tymczasowy:
rm -rf $TMPDIR
```

Dokumentacja pakietu Lmod wykorzystywanego na klastrze Prometheus oraz Zeus: https://kdm.cyfronet.pl/portal/Prometheus:Lmod, http://lmod.readthedocs.io oraz https://www.tacc.utexas.edu/research-development/tacc-projects/lmod/user-guide

Uwagi

- Zaleca się ładować moduły jedynie w skryptach obliczeniowych, a nie w skryptach uruchamianych przy logowaniu na maszynę dostępową lub węzeł obliczeniowy. Dzięki temu łatwiej kontrolować środowisko uruchomieniowe oprogramowania, które ma zostać użyte oraz uniknąć konfliktów pomiędzy załadowanymi modułami.
- Komenda `module avail <nazwa>` znajduje jedynie moduły zaczynające się od ciągu znaków `<nazwa>` dlatego, jeżeli poszukuje się modułów zawierających gdziekolwiek ciąg `<nazwa>` należy wykonać polecenie `module spider <nazwa>`

Wykorzystanie własnych modułów (Zaawansowane)

Pakiet *Lmod* umożliwia również tworzenie własnych modułów (TCL lub Lua). W tym celu należy wykonać polecenie:

```
module use /path/to/personal/modulefiles
```

Podana ścieżka określa katalog w którym użytkownik przetrzymuje własne moduły. Program lmod automatycznie sprawdzi nowe zainstalowane moduły i doda je do listy dostępnych. W modułach można wykorzystywać funkcje języków TCL lub Lua.

Przykładowy moduł w LUA

```
-- -*- lua -*-
local pkgName     = myModuleName()
local fullVersion = myModuleVersion()

whatis("Name: "..pkgName)
whatis("Version "..fullVersion)
whatis("Category: apps/chemistry")
whatis("Description: Molcas@UU 8.0 (v.15-06-18) quantum chemistry chemistry package")
whatis("URL: http://www.kvant.kemi.uu.se/molcas/index.html")
whatis("Keyword: QC, chemistry")

-- add path to Molcas to PATH environmental variable and set MOLCAS environmental variable
prepend_path('PATH','/net/archive/groups/plg******/software/molcas.uu.8/molcas-8.0-15.06.18_CentOS_7.
0_x86_64/bin')
setenv('MOLCAS', '/net/archive/groups/plg******/software/molcas.uu.8/molcas-8.0-15.06.18_CentOS_7.0_x86_64')
add_property("state","testing")

require "math"

--  set Molcas scratch dir accordingly to SLURM environment
if (os.getenv('SLURM_JOB_ID') ~= nil) then
  setenv('MOLCAS_WORKDIR', os.getenv('SCRATCHDIR'))
else
  setenv('MOLCAS_WORKDIR', os.getenv('SCRATCH'))
end
```