Environnement module dans le HPC

CargoDay 7
13 juin 2017, Rennes

CENTRALE NANTES SUPER COMPUTING

Sommaire

- Présentation
- Comment c'était avant ?
- Et maintenant ?
- Retour sur le HPC
- Évolutions
- Conclusion

Présentation

Les modules donnent la possibilité à l'utilisateur de modifier son environnement ($PATH, $LIBRARY_PATH...) à la volée.

Cela permet d'utiliser un soft comme s'il était installé en natif sur le système. On peut aussi naviguer entre plusieurs versions d'un même soft.

On gère les installations, avec les pré-requis de chaque soft sans impacter le système.

Comment c'était avant ?

```
$ cat /etc/redhat-release
CentOS 5.11
$ gcc --version
gcc 4.1.2
```

Dommage, on avait besoin d'un gcc 4

Yum update. On a quand même peu de chance de trouver la bonne version dans les dépôts.

Plusieurs solutions existent :

- yum update
- On récupère le tar.gz, et on fait l'ins
- On upgrade CentOS vers la 6.9 (gcc 4.4.7 par défaut)

L'archive pour l'installation manuelle, pourquoi pas, mais si on souhaite réutiliser l'ancienne version, c'est pas terrible.

Upgrader CentOS, il faut tout refaire...

Et maintenant ?

On a parlé module ? On va tester.
On oublie ce qu'on faisait avant et on installe un paquet.

```
$ yum -y install environment-modules
```

Et comment ça fonctionne ?

Et maintenant ?

```
$ module available
-------------------------------------------------------------------------
/usr/share/Modules/modulefiles
-------------------------------------------------------------------------

$ echo $MODULEPATH
/usr/share/Modules/modulefiles:\
/etc/modulefiles:\
/usr/share/modulefiles
```

Au début, il n'y a rien de configuré bien sûr, mais on sait où se trouvent les modulefiles.

Et maintenant ?

Avant de créer un modulefile, il faut installer l'application souhaitée.
Donc on prend notre tar.gz de gcc 4.4.7.

```
$ ./configure –-prefix=/usr/local/gcc/4.4.7
$ make
$ make install
$ /usr/local/gcc/4.4.7/bin/gcc –-version
gcc 4.4.7
```

Et maintenant ?

Le modulefile associé

```
$ vi /usr/share/Modules/modulefiles/gcc/4.4.7
#%Module 1.0
#
#  gcc 4.4.7

proc ModulesHelp { } {
        puts stderr "\tAdd gcc 4.4.7 in your environment"
}

module-whatis "
Software --------> Compilateur GNU"

prepend-path    PATH            /usr/local/gcc/4.4.7/bin
prepend-path    LIBPATH          /usr/local/gcc/4.4.7/lib
prepend-path    MANPATH         /usr/local/gcc/4.4.7/share/man
prepend-path    INCLUDE           /usr/local/gcc/4.4.7/include
```

Et maintenant ?

Quelques commandes de base :

```
$ module available
------------------------------------------------------------------------------
/usr/share/Modules/modulefiles
------------------------------------------------------------------------------
gcc/4.4.7
$ module help gcc/4.4.7
---------- Module Specific Help for 'gcc/4.4.7' -------------

      Add gcc 4.4.7 in your environment
$ module whatis gcc/4.4.7
Gcc/4.4.7       :
Software --------> Compilateur GNU
```

Et maintenant ?

Comment utiliser le module ?

```
$ module load gcc/4.4.7
$ module list
Currently Loaded Modulefiles:
  1) gcc/4.4.7
$ which gcc
/usr/local/gcc/4.4.7/bin/gcc
$ echo $PATH
/
usr/local/gcc/4.4.7/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:
/usr/bin
$ gcc --version
gcc 4.4.7
$ cat /etc/redhat-release
CentOS 5.11
```

C’est gagné ! On peut utiliser la version de base (4.1.2), ainsi que la version 4.4.7 avec le module.

Retour sur le HPC

Nous avons vu un exemple pour une application. Mais dans le monde HPC, il y a tout un tas d'applications.

```
-----/usr/share/Modules/modulefiles/compilers
gcc/4.9.3       intel/2015.3.187

-----/usr/share/Modules/modulefiles/libraries
arpack/96/gcc-4.9.3                hdf5/1.8.15/gcc-4.9.3              med/3.0.2
pygobject/2.21.3                   boost/1.62.0/gcc-4.9.3             hdf5/1.8.15/intel-2015.3.187
metis/5.1.0                        qhull/2015.2/gcc-4.9.3             bzip2/1.0.6
mfront/3.0.0/gcc-4.9.3             scotch/6.0.4                       cuda/8.0.44
      swig/3.0.10/gcc-4.4.7              curl/7.52.1                  pcre/8.40
xz/5.2.3                           dolfin/2016.2.0                    iconv/1.14
petsc/3.7.3                        zlib/1.2.11                        eigen/3.3.0/gcc-4.9.3
lapack/3.5.0/gcc/4.9.3             petsc/intelmpi-5.0.3/3.1-p8         zlib/1.2.8/gcc-4.9.3
glibc/2.16.0/gcc-4.9.3              lapack/3.6.1/gcc/4.9.3             petsc/intelmpi-5.0.3/3.1-p8-debug


-----/usr/share/Modules/modulefiles/applications
aster/11.0.10    finemarine/5.1    freefem++/3.48    iciplayer/1.2    lsdyna/9.0.1    openfoam/2.4.0    R/3.3.2
 starccm+/11.04.012    castem/2016    foam-extend/3.2    freefem++/3.49.1    lsdyna/7.1.3    matlab/2016b
openfoam/3.0.1    starccm+/10.06.010

-----/usr/share/Modules/modulefiles/tools
cmake/2.8.9/gcc/4.4.7   cmake/3.2.3/gcc/4.4.7   intelmpi/5.0.3.048    openmpi/1.8.4-gcc
python/2.7.10/gcc/4.4.7 python/2.7.12/gcc/4.9.3 python/3.5.2/gcc/4.9.3  singularity/2.3  cmake/3.1.0/gcc/4.4.7
cmake/3.7.1/gcc/4.9.3   openmpi/1.6.5-gcc      openmpi/1.8.4-intel-mlx python/2.7.12/gcc/4.4.7
python/3.4.3/gcc/4.4.7  singularity/2.2.99     valgrind/3.11.0
```

Retour sur le HPC

On est content de notre trouvaille, tout fonctionne à merveille. Mais ça ne peut pas durer ainsi.

Parce qu'un jour il faut :

1. Installer un soft très récent sur un système ancien
2. Mettre à jour l'OS du calculateur
3. Changer les processeurs pour upgrader la machine

Dans les cas cités, on va devoir réinstaller tous nos softs. Oui, même ceux avec lesquels on a passé beaucoup de temps. On a de la doc, tout va bien, mais on n'a plus de 50 softs quand même…

Deux possibilités s'offrent à nous :

1. On réinstalle tout, on reteste tout, on a un peu de temps à passer...
2. On cherche une autre solution...

Évolutions

Depuis quelques temps, on entend des noms de technologies par-ci par-là. Du style Docker, LXC, Shifter, Singularity. Bref, des conteneurs.
En regardant de plus près, on se rend compte que ça peut embarquer des applications pour les faire tourner sur n'importe quel OS.

C'est pas mal, mais le HPC reste spécifique.
On a de l'Infiniband, du MPI, des GPUs, un gestionnaire de batch…
Certains softs pour les conteneurs ne sont pas encore au top pour le HPC, mais Singularity sort du lot avec :

- La gestion de MPI
- Accès IB natif
- Accès GPU natif
- Compatibilité Docker
- Lancement des calculs via SLURM

Évolutions

- Les conteneurs Singularity sont créés et installés par root, ou par un utilisateur avec des droits sudo.
- Une fois installé, un conteneur est toujours en read only, sauf avec l'option -w
- On peut monter des répertoires pour écrire des données.

```
CONTAINER USAGE COMMANDS:
    exec        Execute a command within container
    run         Launch a runscript within container
    shell       Run a Bourne shell within container
    test        Execute any test code defined within container
```

Évolutions

```
ubuntu_w_TFlow.def
BootStrap: debootstrap
OSVersion: xenial
MirrorURL: http://us.archive.ubuntu.com/ubuntu/
%setup
    # commands to be executed on host outside container during bootstrap
%post
    # commands to be executed inside container during bootstrap
    # add universe repo to apt sources and install some packages
    sed -i 's/$/ universe/' /etc/apt/sources.list
    apt-get -y update
    apt-get -y install vim wget python python-pip git
    # install tensorflow
    pip install --upgrade pip
    pip install tensorflow
    # grab the models
    cd /
    git clone https://github.com/tensorflow/models.git
%runscript
    # commands to be executed when the container runs
%test
    # commands to be executed within container at close of bootstrap process
```

Conclusion

Le quotidien des administrateurs HPC est facilité par les conteneurs, mais les modules restent utiles.

- Modules
  - Installation manuelle de chaque soft
  - Facile à configurer
  - Tout recompiler en cas de changement de calculateur ou d'OS

- Conteneurs
  - Un fichier de configuration pour chacun
  - Portable
  - Tester pour comparer les performances avec l'usage "classique"

Environnement module dans le HPC

Merci pour votre attention.

Questions ?