TP easybuild

Christophe Pera
Geoffroy Berret
FLMSN
(Fédération Lyonnaise de Modélisation des Sciences Numériques)
Université de Lyon

Qu’est ce qu’easybuild

- Compilation et installation automatique
  - Gestion automatique des dependances.
  - Gestion automatique des modules(Tcl or Lua syntax)
- Traces complètes (logs) des opérations
- Achives des builds
- Configurables par des fichiers, les variables d’environnement, la ligne de commande
- Extensible et dynamique (easyblocks, toolchains, etc)

Qu’est ce qu’easybuild

- Une interface uniforme
- Automatise les taches repetitives ...
- Offre une pile logiciel coherente
- Expertises pour les logiciels HPC
- EasyBuild 3.6.0 supports 1477 different software packages (incl. toolchains, bundles):
  - http://easybuild.readthedocs.io/en/latest/version-specific/Supported_software.html
  - compilers: GCC,Intel, Clang, PGI, IBM XL, Cray
  - MPI libraries:OpenMPI, Intel MPI, MPICH, MPICH2, MVAPICH2, ...
  - BLAS/LAPACK libraries:Intel MKL, OpenBLAS

Terminologie

- Le framwork EasyBuild
  - core of EasyBuild: Python modules& packages
  - Offre les fonctions de compilaiton et d'installation
- easyblock
  - Un module python « build script », 'plugin' pour le framework
  - Implements une procedure generique pour la compilation, l'installation, … du logiciel.
- Easyconfig (*.eb): specifie les parammetres de la generation du logiciel; software name/version, compiler toolchain, etc.
- (compiler) toolchain: defini le compiliateur et les librairies necessaires (MPI, BLAS/LAPACK, ...)

Pre-requis

- Orienté Linux x86_64HPCsystems
- Distibutions RedHat-based (CentOS, SL, ...), ou Debian et SuSE (ou leurs derivés)
- Peut fonctionner sur macOS ….
- Support basique pour Windows … mais est ce necessaire ?
- Des evolutions pour supporter AARCH64 (ARM) et PowerLinux
- Python 2.6 ou version ulterieureormore , incl. setuptools (pas de support Python 3)
- Compilateur CC++ system (bootstrap)
- Un logiciel « module » (Lmod is highly

Installation

```
# download bootstrap script from
# https://raw.github.com/hpcugent/easybuild-framework/develop/easybuild/scripts/bootstrap_eb.py
$ python bootstrap eb.py $PREFIX
$ module use $PREFIX/modules/all
$ module load EasyBuild
```

- Quelques dépendances (exemple Centos 7.4):
  - python-setuptools GitPython pysvn graphviz Lmod bzip2 patch gcc g++ unzip ...
  - Pour openmpi : Libibverbs-devel
  - Pour Python : openssl-devel

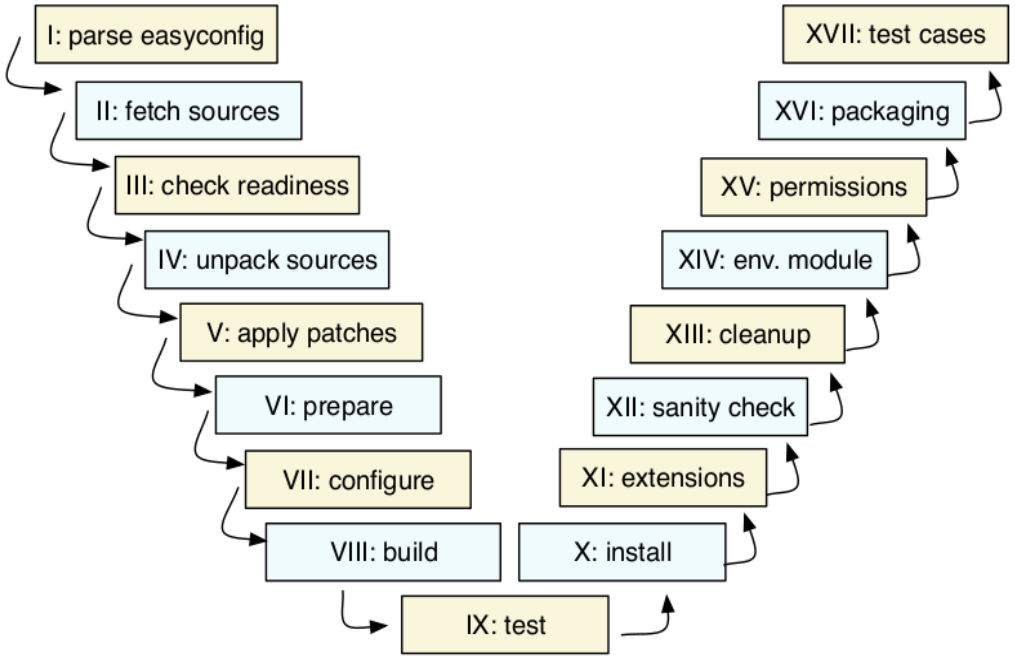
Configuration

- Des fichiers : key-value lines, text files (e.g., prefix=/tmp)
- Des variables d'environnement (e.g., $EASYBUILD_PREFIX set to /tmp)
- Des paramètres de ligne de commande (e.g., --prefix=/tmp)

```
$ EASYBUILD_PREFIX=/tmp eb --buildpath /dev/shm --show-config
#
# Current EasyBuild configuration
# (C: command line argument, D: default value, E: environment variable, F: configuration file)
#
buildpath      (C) = /dev/shm
installpath    (E) = /tmp
packagepath    (E) = /tmp/packages
prefix         (E) = /tmp
repositorypath (E) = /tmp/ebfiles_repo
robot-paths    (D) = /Users/kehoste/work/easybuild-easyconfigs/easybuild/easyconfigs
sourcepath     (E) = /tmp/sources
```

Le process ...

EasyBuild performs a step-wise installation procedure for each software



- download sources (best effort)
- set up build directory & environment
  - unpack sources (& apply patches)
  - load modules for toolchain & deps
  - define toolchain-related env vars (`$CC`, `$CFLAGS`, ...)
- configure, build, (test), install, (extensions)
- perform a simple sanity check on installation
- generate module file

each step can be customised via easyconfig parameters or an easyblock

Easyconfig

software name and version

build variant (specific to WRF)
('dmpar': distributed, MPI)

software metadata

toolchain name & version

sources & patches

list of (build) dependencies
note: all versions are *fixed*!

```
name = 'WRF'
version = '3.8.0'

buildtype = 'dmpar'  # custom parameter for WRF
versionsuffix = '-' + buildtype  # part of module name

homepage = 'http://www.wrf-model.org'
description = "Weather Research and Forecasting (WRF) Model"

toolchain = {'name': 'intel', 'version': '2016b'}

source_urls = ['http://www.mmm.ucar.edu/wrf/src/']
sources = ['%(name)sV%(version_major_minor)s.TAR.gz']
patches = ['WRF-%(version)s_known_problems.patch']

builddependencies = [('tcsh', '6.20.00')]
dependencies = [
    ('JasPer', '2.0.10'),
    ('netCDF', '4.4.1'),
    ('netCDF-Fortran', '4.4.4'),
]
```

Easyblock : règles de build génériques

- ConfigureMake
  standard ' ./configure ' - ' make ' - ' make install ' installation procedure
- CMakeMake
  Presque comme ConfigureMake, mais basé sur CMake pour la configuration
- PythonPackage
  Installe des paquets Python (' python setup.py install ', ' pip install ', ...)
- MakeCp
  Pour les configuration non standard, build basé sur ' make ', realise les installation par copie de binaires et librairies
- Tarball
  extrait les archives et copie
- Binary
  execute un code binaire pour l’installation

Usages

- Utiliser la collection de logiciel déjà intégrée dans Easybuild
- Étendre et ajouter un nouveau logiciel, une nouvelle version, ...

Intégrer un nouveau logiciel : idées

- Définir les paramètres easyconfig pour spécifier l'installation (on peut se baser sur un fichier .eb existant)
- Pour un modification de toolchain, essayer eb –try-*
  - Pour une version logiciel differente: --try-software-version
  - Pour une version de toolchain differente : --try-toolchain
  - --try-* options prennent en compte --robot
- Pour un nouvelle procédure standard d'installation, réutiliser un easyblock existant doit etre réutiliser

Les toolchains

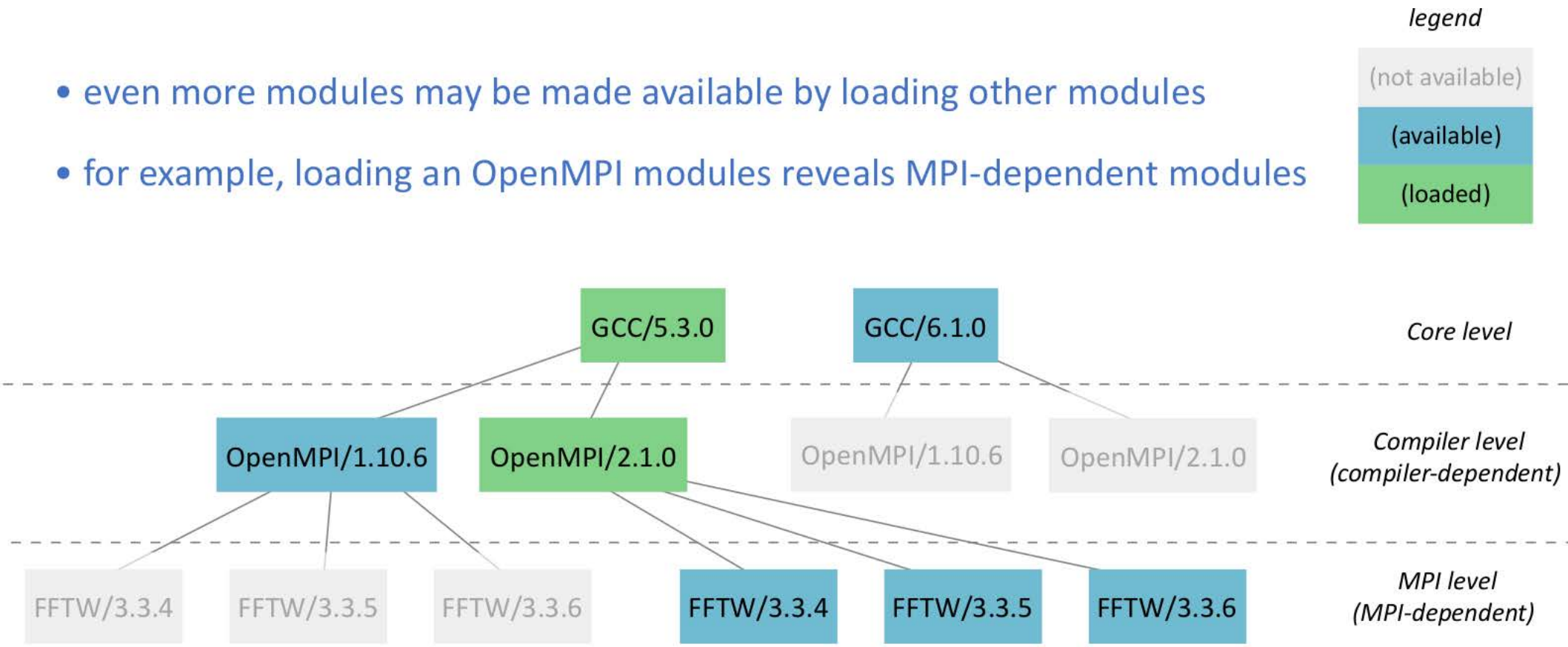
- http://easybuild.readthedocs.io/en/latest/Common-toolchains.html
- intel et foss toolchains sont les plus utilisés.
- Mise à jour semestrielle (2016b , 2017a , ...)
- Les versions « stables
  - foss/2017a
    - binutils 2.27, GCC 6.3.0, OpenMPI 2.0.2, OpenBLAS 0.2.19, LAPACK 3.7.0, FFTW 3.3.6
  - intel/2017a
    - binutils 2.27 + GCC 6.3.0 as base
    - version 2017.1.132 du compilateurs Intel , MPI et Intel MKL
- Les dernières versions : foss/2018a et

Template de nommage des modules

- eb --avail-module-naming-schemes
- On peut choisir/imposer ses regles via --module-naming-scheme
- default: EasyBuildMNS (<name>/<version>-<toolchain>-<versionsuffix>)

Exemple : modèle hiérarchique

- even more modules may be made available by loading other modules
- for example, loading an OpenMPI modules reveals MPI-dependent modules



Difficultés

- EasyBuild est un ensemble de contributions non parfaites (pas de validation stricte comme pour des paquets de distribution)
- Les difficultés rencontrées frequement sont :
  - failure to download source tarballs/installation files (solution : gerer son propre depot … ou recuperer l'archive manuellement et la stocker dans le rep sources)
  - build failures de logicie:w
  - ls anciens sur des systèmes recents.
    - Ne pas utiliser la nouvelle version
    - Consulter l'expert par l'interface « google »
  - Dépeenandeences manquantes dans la specification de easybuildr

Difficultés … ?

- Multiples versions d’une même logiciel.
- Logiciel de version identique avec des dépendances différentes.
- Le choix de la toolchain et sa compatibilité avec les logiciels souhaités.
- Les dépendances riches qui imposent des temps de « build » très long.
- Difficultés d’organisations, de gestion des logiciels par rapport aux toolchains, versions, instructions (illegal instructions si matériels hétérogènes, compilation partielle insidieuse en fonction des specifications/dépendances comme pour python …).

Gcc-4.7.2.eb

```
name = "GCC"
version = '4.7.2'

homepage = 'http://gcc.gnu.org/'
description = "The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran,
                    Java, and Ada, as well as libraries for these languages (libstdc++, libgcj,...)."

toolchain = {'name':'dummy', 'version': 'dummy'}
```

Intgration GitHub, aide et Contribution

- Intégration d’outils et de commande pour contribuer via github au projet easybuild
  - Eb --new-pr file.eb
  - Eb –from-pr
  - Eb –check-pr
- EasyBuild wiki pour les tutoriaux & guidelines:
  - https://github.com/hpcugent/easybuild/wiki/Contributing-back
  - https://github.com/hpcugent/easybuild/wiki/Policy-for-easyconfig-pull-requests

TP – environnement de travail

- PC personnel (OS linux + connexion internet + python, gcc/g++)
- Serveurs de test (FLMSN)
  - Cluster de test centos7, slurm 17.11, ivybridge
  - Compte formation : ust4hpc[1-20]
  - Nœud de login : p2chpd-login0.univ-lyon1.fr
  - ATTENTION : home user en NFS …
    - VOUS DEVEZ ABSOLUMENT TRAVAILLER DANS /tmp

Bootstrap … 48s ...

```
#!/bin/sh
yum -y install  python-setuptools GitPython pysvn
   graphviz Lmod
yum -y install libibverbs-devel
PREFIX_X86_64=/tmp/eb_ust4hpc
mkdir -p ${PREFIX_X86_64}
cd ${PREFIX_X86_64}
curl -O
   https://raw.githubusercontent.com/hpcugent/eas
   ybuild-
   framework/develop/easybuild/scripts/bootstrap_
   eb.py
```

Usage élémentaire

http://easybuild.readthedocs.org/en/latest/Using_the_EasyBuild_command_line.html

- On specifie nom/versiondu logiciel et toolchain avec la commande ‘eb’
- On utilise le nom complet du fichier easyconfig:
  - eb GCC-4.9.2.eb Clang-3.6.0-GCC-4.9.2.eb
- Ou un repertoire local ou sont localisés les fichiers easyconfig:
  - eb $HOME/myeasyconfigs
- Ou en utilisant la ligne de commande:
  - eb --software-name=GCC
- --robot/-r: dependency resolution, --debug/-d:

Workflow élémentaire

http://easybuild.readthedocs.org/en/latest/Typical_workflow_example_with_WRF.html

- Recherche de l'existence d'un logiciel dans l'existant(case-insensitive):
  - eb -S hpl
- Recuperer le nom identifiant le logiciel, la version et la toolchain souhaité, etc.
- Verifier et tester le build (dry run):
  - eb HPL-2.2-foss-2018a.eb -D
- Activer les logs debug et la résolution de dépendance ('robot' mode):
  - eb HPL-2.2-foss-2018a.eb -dr

Test de l’environnement

[ust4hpc-001@node001 ~]$ echo "export ASYBUILD_PREFIX=\"/dir1/dir2\" ~/.basrc

[ust4hpc-001@node001 ~]$ echo "module use ${EASYBUILD_PREFIX}/modules/all

[ust4hpc-001@node001 ~]$source ~/.bashrc

[ust4hpc-001@node001 ~]$module load EasyBuild

[ust4hpc-001@node001 ~]$eb –version

[ust4hpc-001@node001 ~]$module av

[ust4hpc-001@node001 ~]$module list

[ust4hpc-001@node001 ~]$eb --list-easyblocks

[ust4hpc-001@node001 ~]$eb –ist-toolchains

[ust4hpc-001@node001 ~]$eb --dep-graph=depgraph.png …

[ust4hpc-001@node001 ~]$eb GCC-4.9.2.eb Clang-3.6.0-GCC-4.9.2.eb -Dr

Une première installation M4 quelques minutes ….

[ust4hpc-001@node001 ~]$ eb -S "^M4"

….

M4-1.4.17.eb

….

[ust4hpc-001@node001 ~]$ eb M4-1.4.17.eb -Dr

=> liste 0 dépendances ...

[ust4hpc-001@node001 ~]$ eb M4-1.4.17.eb

[ust4hpc-001@node001 ~]$eb M4-1.4.17.eb -f

à consulter : les traces du terminal, les logs du build, le fichier, M4-1.4.17.eb .

```
[ust4hpc-001@node001 ~]$ eb M4-1.4.17.eb -f
== temporary log file in case of crash /tmp/eb-EgQ_sz/easybuild-qrVUfh.log
== processing EasyBuild easyconfig
/tmp/eb_ust4hpc/software/EasyBuild/3.6.0/lib/python2.7/site-
packages/easybuild_easyconfigs-3.6.0-py2.7.egg/easybuild/easyconfigs/m/M4/M4-
1.4.17.eb
== building and installing M4/1.4.17...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== patching...
== preparing...
== configuring...
== building...
== testing...
== installing...
== taking care of extensions...
== postprocessing...
== sanity checking...
== cleaning up...
== creating module...
== permissions...
== packaging...
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file(s)
/tmp/eb_ust4hpc/software/M4/1.4.17/easybuild/easybuild-M4-1.4.17-
20180516.121201.log
== Build succeeded for 1 out of 1
== Temporary log file(s) /tmp/eb-EgQ_sz/easybuild-qrVUfh.log* have been removed.
== Temporary directory /tmp/eb-EgQ_sz has been removed.
[ust4hpc-001@node001 ~]$
```

Installation R foss 2017 …. 5h

```
[ust4hpc@node002 ~]$ module use /tmp/eb_ust4hpc/modules/
all/   tools/
[ust4hpc@node002 ~]$ module use /tmp/eb_ust4hpc/modules/all

[ust4hpc@node002 ~]$ module load EasyBuild

[ust4hpc@node002 ~]$ export EASYBUILD_PREFIX=/tmp/eb_ust4hpc/

[ust4hpc@no de002 ~]$ echo "export EASYBUILD_PREFIX=/tmp/eb_ust4hpc/"
>> ~/.bashrc
 [ust4hpc@node002 ~]$ echo "module use /tmp/eb_ust4hpc/modules/all" >>
~/.bashrc
[ust4hpc@node002 ~]$ echo "module load EasyBuild" >> ~/.bashrc

[ust4hpc@node002 ~]$ time eb R-3.4.3-foss-2017b-X11-20171023.eb -r

…….    on va éviter de le faire …….
                mais plutôt tester

[ust4hpc@node002 ~]$ time eb R-3.4.3-foss-2017b-X11-20171023.eb -r -Dr

      ===> liste 80 dépendances ...
```

R et openmpi … (on triche)

- Reutiliser les pre-installations dans /tmp/eb_ust4hpc
  - Module purge
  - module use /tmp/eb_ust4hpc/modules/all
  - Export EASYBUILD_PREFIX=/tmp/eb_${USER}
  - module use /tmp/eb_${USER}/modules/all
- La suite, Tp singularity – partie exercice sur R et mpi.

Quelques références ...

- Présentations :
  - https://fosdem.org/2018/schedule/event/contributor_automation/attachments/slides/2272/export/events/attachments/contributor_automation/slides/2272/simplifying_the_contribution_process_fosdem18.pdf
  - http://users.ugent.be/~kehoste/EasyBuild_20170425_PRACE_Spring_School.pdf
- website:
  - http://hpcugent.github.io/easybuild/
- documentation:
  - http://easybuild.readthedocs.io
- Cheat sheet :
  - http://linksceem.cyi.ac.cy/ls2/images/stories/HPCBIOS_cheatsheet.pdf