INSTITUT DU
DÉVELOPPEMENT ET DES
RESSOURCES EN
INFORMATIQUE
SCIENTIFIQUE

www.idris.fr

Introduction à Spack et retour d'expérience de l'IDRIS

L'IDRIS ?

- Un des quatre centres de calcul nationaux
- Unité d'Appui à la Recherche CNRS avec 33 ITA
- Communautés :
  - Calcul Haute Performance
  - Intelligence Artificielle (depuis 2019 : rapport Villani et plan IA For Humanity)
- Jean Zay : Calculateur « convergé » pour usages HPC et IA

Jean Zay

28 Pflop/s
Puissance crête cumulée

2696
GPU NVIDIA V100 16-32 Go

86344 cœurs
de calcul Intel Cascade Lake 2.5 GHz

427 To
de mémoire

100 Gb/s
Réseau d'interconnexion Omni-Path
(1 lien par nœud scalaire et 4 liens par nœud convergé)

2,2 Po 450 Go/s
Full Flash

35 Po 150 Go/s

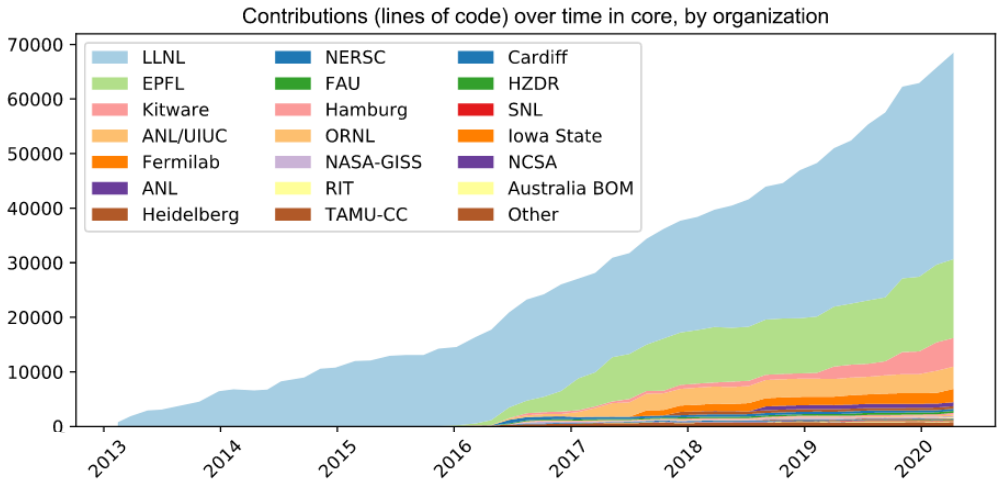
Introduction à Spack

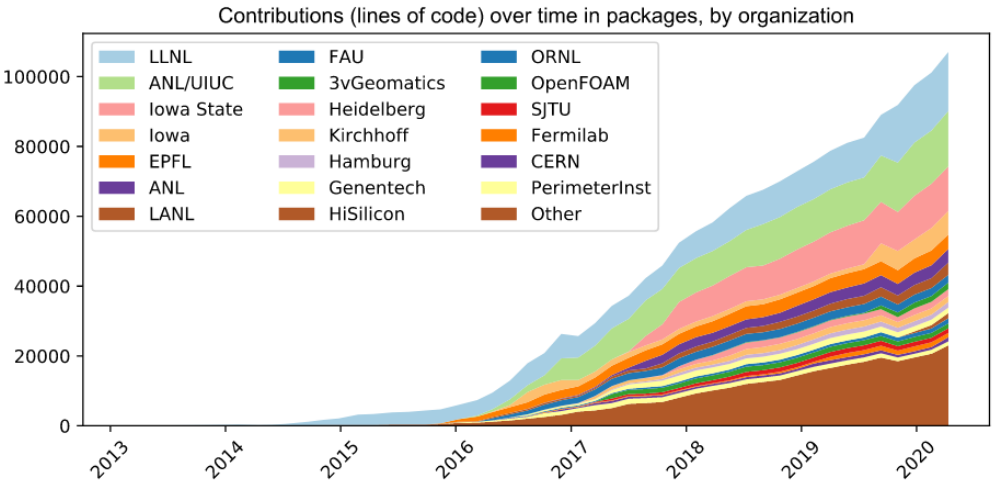
Spack

- **S**upercomputer **PACK**age manager
- Gestionnaire de paquets « à partir des sources »
- Développé au Lawrence Livermore National Laboratory depuis 2013
- Inspiré à l’origine par Homebrew et Nix
- Orientation calcul scientifique et haute performance

Un projet actif

- Très nombreuses contributions, d'origines variées :



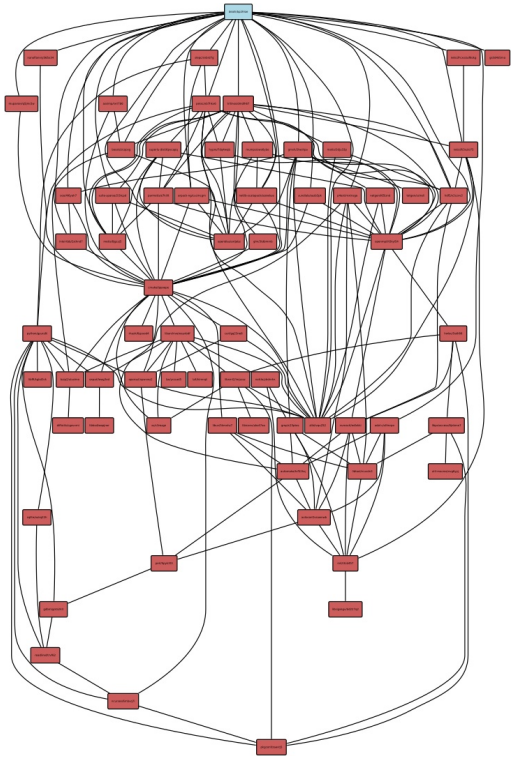


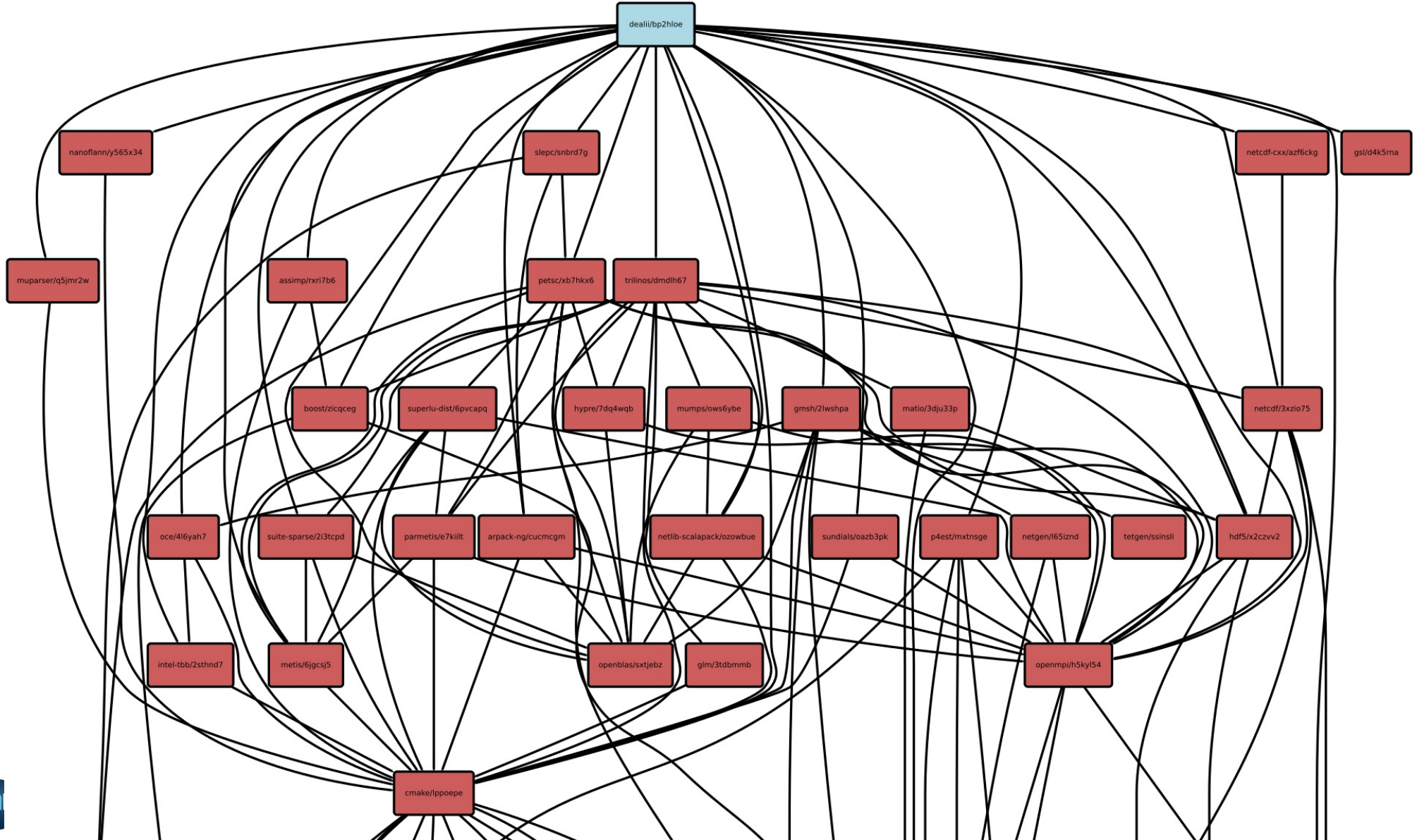
Un projet actif

- Bonne implantation :
  - 3 premières machines au TOP500 :
    - Fugaku (RIKEN, Japon)
    - Summit (ORNL, USA)
    - Sierra (LLNL, USA)
  - Exascale Computing Project

Enfer des dépendances

- Exemple de la bibliothèque Deal.II (éléments finis) :




dealii/bp2hloe
nanoflann/y565x34
slepc/snbrd7g
netcdf-cxx/azf6ckg
gsl/d4k5rna
muparser/q5jmr2w
assimp/rxri7b6
petsc/xb7hkx6
trilinos/dmdlh67
boost/zicqceg
superlu-dist/6pvcapq
hypre/7dq4wqb
mumps/ows6ybe
gmsh/2lwshpa
matio/3dju33p
netcdf/3xzio75
oce/4l6yah7
suite-sparse/2i3tcpd
parmetis/e7kiilt
arpack-ng/cucmcgm
netlib-scalapack/ozowbue
sundials/oazb3pk
p4est/mxtnsge
netgen/l65iznd
tetgen/ssinsli
hdf5/x2czvv2
intel-tbb/2sthnd7
metis/6jgcsj5
openblas/sxtjebz
glm/3tdbmmb
openmpi/h5kyl54
cmake/lppoepe

Explosion combinatoire

- Exemple honnête de Jean Zay :
  - Des dizaines de produits et bibliothèques
  - Trois compilateurs : Intel, PGI, GNU
  - Deux implémentations MPI : Intel MPI, OpenMPI
  - Plusieurs versions de chaque...
  - OpenMP/OpenACC/CUDA

Objectifs de Spack

- Simplifier la gestion des dépendances
- Permettre de changer facilement :
  - les versions
  - le compilateur et ses options
  - les bibliothèques comme MPI, BLAS, LAPACK, …
- Améliorer la reproductibilité des installations
- Sans avoir à sacrifier les performances !

Installation de Spack

- Cloner le dépôt Git suffit

```
$ git clone https://github.com/spack/spack
$ . spack/share/spack/setup-env.sh
```

- Si vous avez un compilateur :

```
$ spack install zlib
```

- Par défaut, installation dans le sous-répertoire « opt »

Compilateurs

- Détectés automatiquement au premier lancement

```
$ spack compilers
==> Available compilers
-- gcc ubuntu18.04-x86_64 ---------------------------------
gcc@7.5.0  gcc@5.5.0  gcc@4.8
```

- Ajoutés à la demande

```
$ spack compiler add          # Cherche les compilateurs dans le PATH

$ spack compiler add /rep/install/compilo # Recherche avec un indice
```

- Fichier de configuration : compilers.yaml

Paquets supportés

- De plus en plus nombreux !

```
$ spack list
==> 5151 packages.
3proxy                          ncview                          py-torch-spline-conv
abduco                          ndiff                           py-torchaudio
abi-compliance-checker          nek5000                         py-torchfile
...
nco                             py-torch-nvidia-apex            zsh
ncompress                       py-torch-scatter                zstd
ncurses                         py-torch-sparse                 zziplib
```

- Filtrage possible

```
$ spack list netcdf
==> 6 packages.
netcdf-c  netcdf-cxx  netcdf-cxx4  netcdf-fortran  parallel-netcdf  py-netcdf4

$ spack list petsc
==> 2 packages.
petsc  py-petsc4py
```

Détails d'un paquet

```
$ spack info netcdf-c
AutotoolsPackage:   netcdf-c

Description:
    NetCDF (network Common Data Form) is a set of software libraries and
    machine-independent data formats that support the creation, access, and
    sharing of array-oriented scientific data. This is the C distribution.

Homepage: http://www.unidata.ucar.edu/software/netcdf

Maintainers: @skosukhin @WardF

Tags:
    None

Preferred version:
    4.7.4      ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-c-4.7.4.tar.gz

Safe versions:
    master     [git] https://github.com/Unidata/netcdf-c.git on branch master
    4.7.4      ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-c-4.7.4.tar.gz
    4.7.3      ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-c-4.7.3.tar.gz
...
    4.3.3      ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4.3.3.tar.gz

[A suivre à la prochaine diapositive]
```

Versions connues par Spack

Détails d'un paquet

```
$ spack info netcdf-c [Suite et fin]
Variants:
    Name [Default]           Allowed values    Description
    ====================     ==============    ================================================

    dap [off]                on, off           Enable DAP support
    hdf4 [off]               on, off           Enable HDF4 support
    jna [off]                on, off           Enable JNA support
    mpi [on]                 on, off           Enable parallel I/O for netcdf-4
    parallel-netcdf [off]    on, off           Enable parallel I/O for classic files
    pic [on]                 on, off           Produce position-independent code (for shared libs)
    shared [on]              on, off           Enable shared library

Installation Phases:
    autoreconf    configure    build    install

Build Dependencies:
    autoconf  automake  curl  hdf  hdf5  libtool  m4  mpi  parallel-netcdf  zlib

Link Dependencies:
    curl  hdf  hdf5  mpi  parallel-netcdf  zlib

Run Dependencies:
    None

Virtual Packages:
    None
```

Les options

Les différentes dépendances

Base de données

- Conserve la trace de toutes les installations

```
$ spack find
==> 2737 installed packages
-- linux-rhel7-x86_64 / gcc@4.8.5 ------------------------------
antlr@2.7.7             gcc@8.2.0         libgeotiff@1.5.1      libxtst@1.2.2           python@2.7.16
apr@1.6.2               gcc@8.3.0         libgit2@0.26.0        lua@5.3.5               qt@5.8.0
...

-- linux-rhel8-x86_64 / intel@19.1.1 ---------------------------
elsi@2.2.1       elsi@2.5.0      elsi@2.6.2         hdf5@1.10.5        hypre@2.19.0      zlib@1.2.11

-- linux-rhel8-x86_64 / pgi@20.1 -------------------------------
hwloc@2.0.2      metis@5.1.0     openmpi@4.0.2      parmetis@4.0.3     slurm@18-08-0-1
```

- Possibilité
  - d’avoir plusieurs installations d’un même produit
  - de gérer des architectures différentes

Base de données

- Arbre des dépendances

```
$ spack find -d netcdf-c
...
-- linux-rhel8-skylake_avx512 / intel@19.1.1 --------------------
netcdf-c@4.7.4
    hdf5@1.12.0
        zlib@1.2.11

netcdf-c@4.7.4
    hdf5@1.12.0
        numactl@2.0.12
        openmpi@4.0.5
            hwloc@2.2.0
                libpciaccess@0.16
                libxml2@2.9.10
                    libiconv@1.16
                    xz@5.2.5
                    zlib@1.2.11
            opa-psm2@11.2.166
            slurm@18-08-0-1
...
```

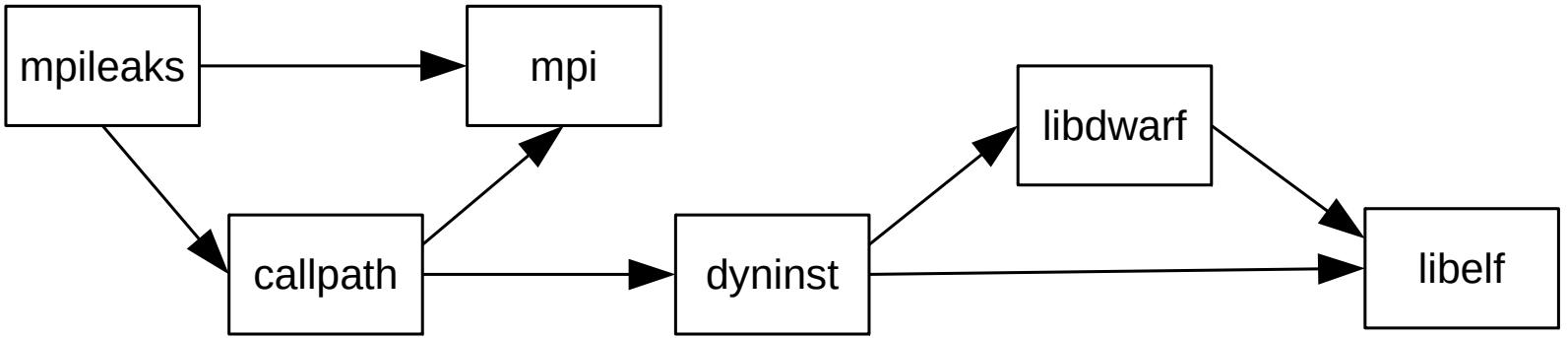
Langage de spécification

- Grande force de Spack

```
$ spack install netcdf-c                                  # Aucune contrainte
$ spack install netcdf-c@4.7.3                            # Version spécifique
$ spack install netcdf-c%gcc@8.3.1                        # Compilateur spécifique
$ spack install netcdf-c+mpi~hdf4                         # Variantes spécifiques
$ spack install netcdf-c ^openmpi                         # Dépendance spécifique
$ spack install netcdf-c cflags="-O3 -g" ldflags="-O3 -g" # Options de compilation spécifiques
$ spack install netcdf-c@4.7.3%gcc@8.3.1 ^hdf5@10.1.7+hl # On peut combiner toutes les
                                                          # possibilités récursivement !
```

Graphe de dépendances

- Une installation = un graphe orienté acyclique



- Cohérence garantie :
  - Pas de mélange de versions dans les dépendances
  - Mélange de compilateurs autorisés (mais prudence)

Graphe de dépendances

- Un graphe de dépendances = une empreinte unique

  ⇒ Tout est pris en compte (versions, variantes, etc)

- Empreinte utilisée dans les répertoires d'installation

  ⇒ Solution à l'explosion combinatoire

```
opt
└──spack
   └──linux-ubuntu18.04-broadwell
      └──gcc-7.5.0
         ├──hdf5-1.10.7-5umosquucqcsgeq6l56bpyeze5lr5y5l
         ├──hdf5-1.10.7-lc2vgm75g5avg3enp6pi6wygrkdkmoe4
         ├──libsigsegv-2.12-cy6wvhkposdmsbcu6vziegcxyxnlrmir
         ├──libszip-2.1.1-m7btczwncfg4xkgvh55itdrqea3om22o
         ├──m4-1.4.18-ddc3vhixyli3j7x32iovn2hrbqhfmsg5
         ├──netcdf-c-4.7.4-az3aojtywmb5yntsx6rotgdgnqf55zot
         └──zlib-1.2.11-x6rxxaqcl3lgzvkl7qwzc6pkkmxfjdkh
```

Empreinte et base de données

- Partie intégrante de la base de données

```
$ spack find -lv hdf5
...
-- linux-ubuntu18.04-broadwell / gcc@7.5.0 --------------------
lc2vgm7 hdf5@1.10.7~cxx~debug~fortran~hl~java~mpi+pic+shared~szip~threadsafe api=none
5umosqu hdf5@1.10.7~cxx~debug~fortran+hl~java~mpi+pic+shared+szip~threadsafe api=none
```

- Référence possible à une empreinte

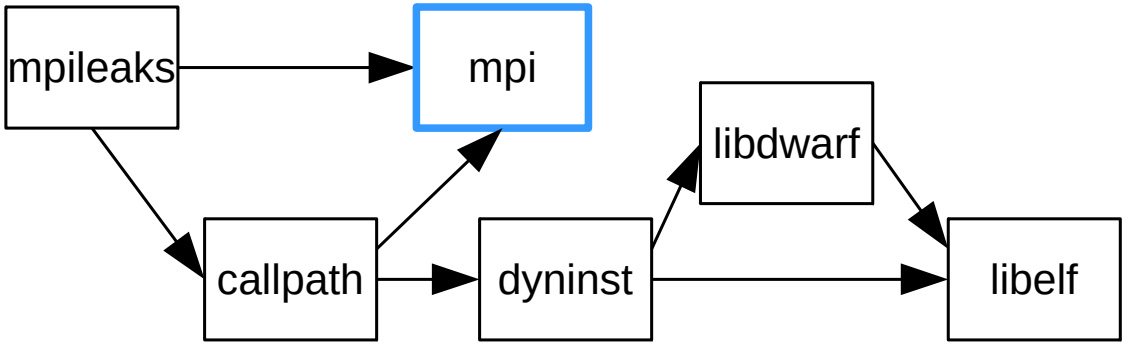
```
$ spack find -Lv /x6rxxa
...
-- linux-ubuntu18.04-broadwell / gcc@7.5.0 --------------------
x6rxxaqcl3lgzvkl7qwzc6pkkmxfjdkh zlib@1.2.11+optimize+pic+shared

$ spack install netcdf-c ^/x6rxxa
```

Dépendances virtuelles

- Bibliothèques avec :
  - la même API (potentiellement versionnée)
  - mais pas la même ABI

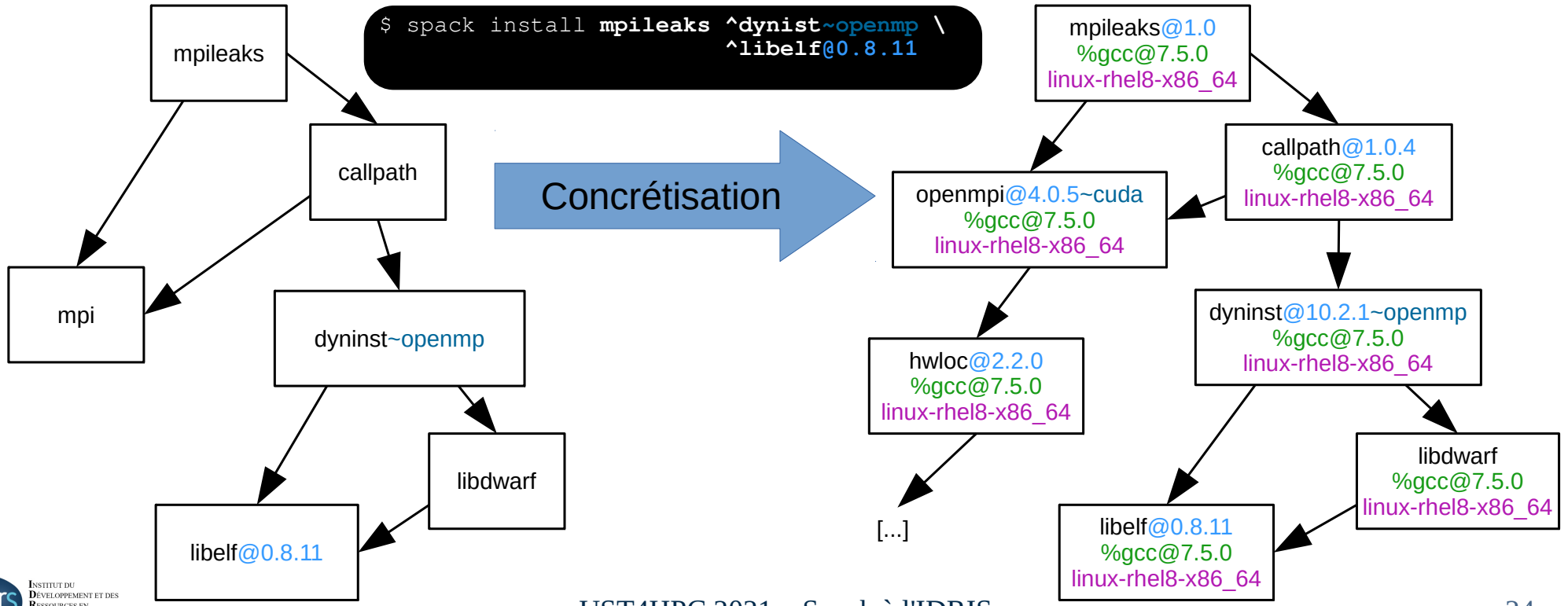
⇒ Solution élégante pour gérer MPI, LAPACK, etc




```
$ spack install mpileaks ^intel-mpi
$ spack install mpileaks ^openmpi
$ spack install mpileaks ^mpi@3
```

Concrétisation

- Complète la spécification abstraite fournie par l'utilisateur



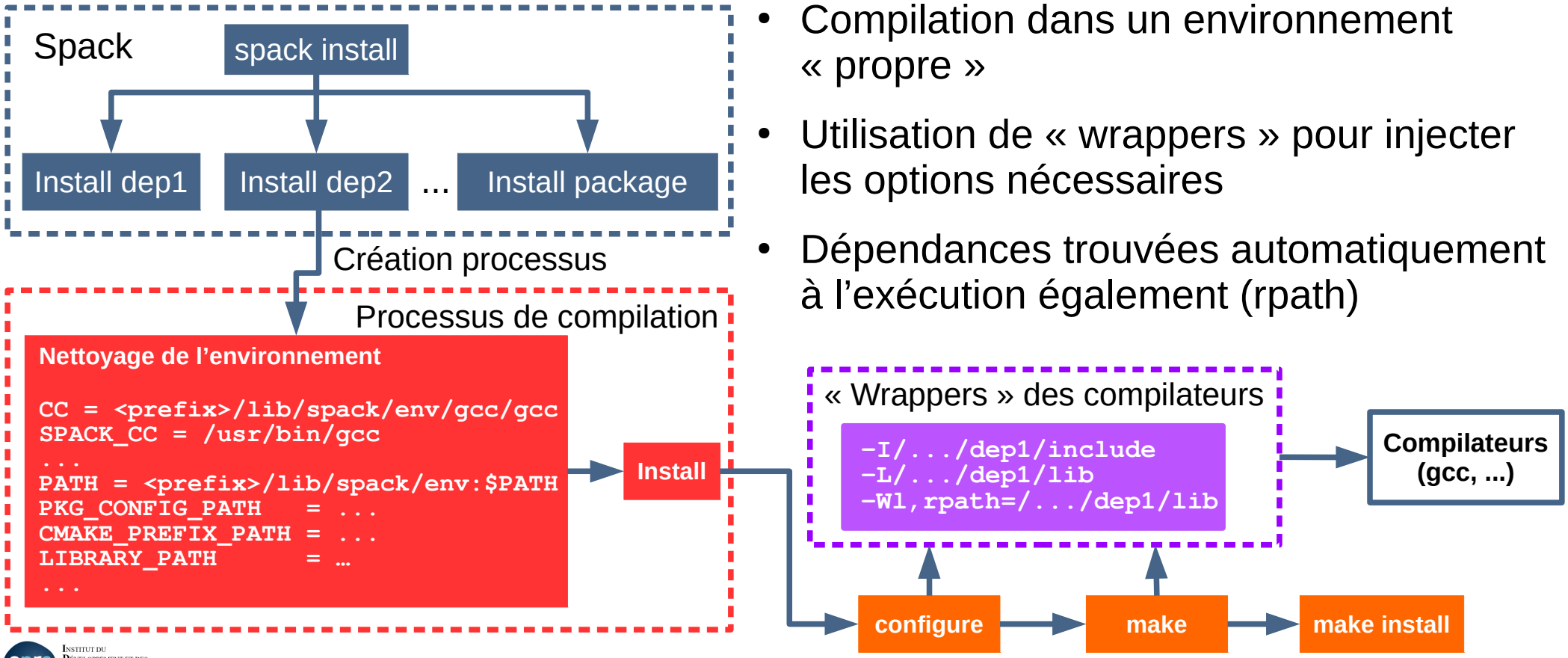
Concrétisation

- Possibilité de la visualiser avant l'installation

```
$ spack spec -It netcdf-c~mpi ^hdf5~mpi+szip
Input spec
--------------------------------
 -   [    ]  netcdf-c~mpi
 -   [    ]      ^hdf5~mpi+szip

Concretized
--------------------------------
 -   [    ]  netcdf-c@4.7.4%gcc@7.5.0~dap~hdf4~jna~mpi~parallel-netcdf+pic+shared arch=linux-ubuntu18.04-
broadwell
 -   [bl  ]      ^hdf5@1.10.7%gcc@7.5.0~cxx~debug~fortran+hl~java~mpi+pic+shared+szip~threadsafe api=none
arch=linux-ubuntu18.04-broadwell
[+]  [bl  ]          ^libszip@2.1.1%gcc@7.5.0 arch=linux-ubuntu18.04-broadwell
[+]  [bl  ]          ^zlib@1.2.11%gcc@7.5.0+optimize+pic+shared arch=linux-ubuntu18.04-broadwell
[+]  [b   ]      ^m4@1.4.18%gcc@7.5.0+sigsegv arch=linux-ubuntu18.04-broadwell
[+]  [bl  ]          ^libsigsegv@2.12%gcc@7.5.0 arch=linux-ubuntu18.04-broadwell
```

Mécanisme de compilation



- Compilation dans un environnement « propre »
- Utilisation de « wrappers » pour injecter les options nécessaires
- Dépendances trouvées automatiquement à l'exécution également (rpath)

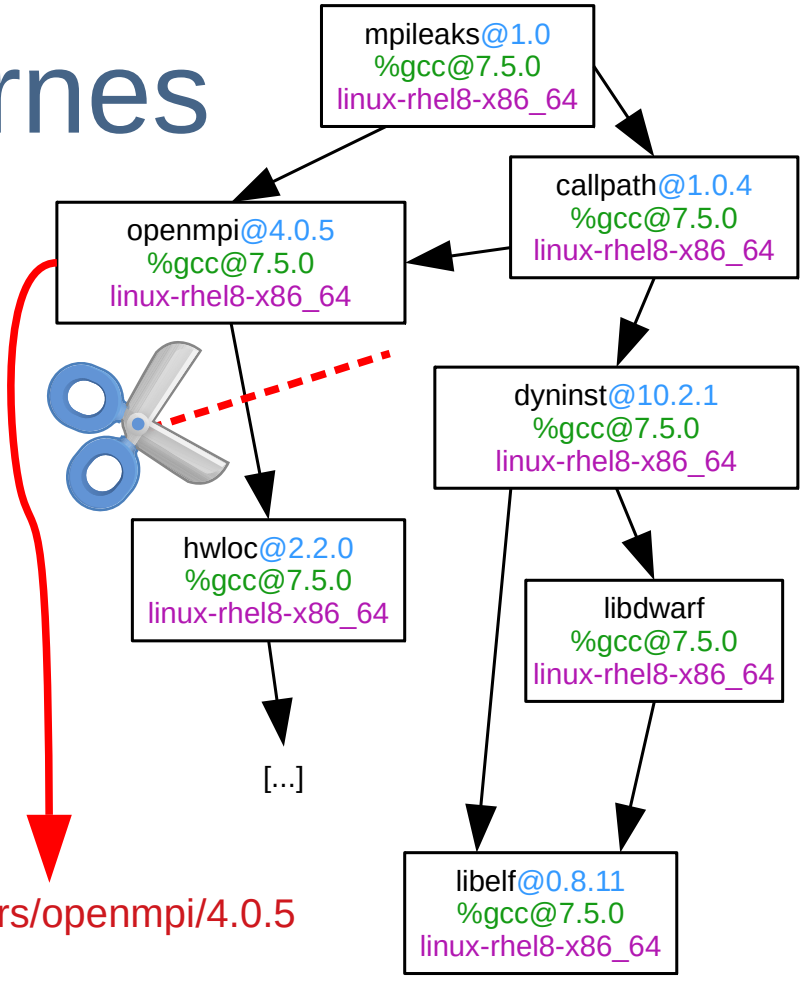
Paquets externes

- « Coupe » le graphe pour pointer vers une ressource externe

```
$ spack install mpileaks ^openmpi@4.0.5
```

packages.yaml :

```
packages:
  openmpi:
    buildable: False
    externals:
    - spec: openmpi@4.0.5~cuda
      prefix: /chemin/vers/openmpi/4.0.5
    - spec: openmpi@4.0.5+cuda
      prefix: /chemin/vers/openmpi/4.0.5-cuda
    - spec: openmpi@4.1.0~cuda
      prefix: /chemin/vers/openmpi/4.1.0
```



Optimisations

- Détection de l’architecture

```
$ spack arch
linux-rhel8-skylake_avx512
```

- Conversion en options de compilation
  Ex : **`-march=skylake-avx512 -mtune=skylake-avx512`**
- Injection via les « wrappers » des compilateurs

Utiliser les produits installés

- Directement via Spack :

```
$ which ncdump

$ spack load netcdf-c # N'importe quelle spécification

$ which ncdump
/.../spack/opt/spack/.../netcdf-c-4.7.4-az3aojtywmb5yntsx6rotgdgnqf55zot/bin/ncdump
```

Utiliser les produits installés

- Via les modules générés automatiquement par Spack :

```
#%Module1.0
## Module file created by spack (https://github.com/spack/spack) on 2021-01-13 11:21:52.160880
##
## netcdf-c@4.7.4%gcc@7.5.0~dap~hdf4~jna~mpi[...] arch=linux-ubuntu18.04-broadwell/az3aojt
##
## Configure options: --enable-v2 --enable-utilities [...]
##

module-whatis "NetCDF (network Common Data Form) is a set of software libraries and machine-independent
data formats that support the creation, access, and sharing of array-oriented scientific data. This is the
C distribution."
[...]
prepend-path PATH ".../spack/opt/spack/.../netcdf-c-4.7.4-az3aojtywmb5yntsx6rotgdgnqf55zot/bin"
prepend-path LD_LIBRARY_PATH ".../spack/opt/spack/.../netcdf-c-4.7.4-az3aojtywmb5yntsx6rotgdgnqf55zot/lib"
prepend-path LIBRARY_PATH ".../spack/opt/spack/.../netcdf-c-4.7.4-az3aojtywmb5yntsx6rotgdgnqf55zot/lib"
[...]
prepend-path CMAKE_PREFIX_PATH ".../spack/opt/spack/.../netcdf-c-4.7.4-az3aojtywmb5yntsx6rotgdgnqf55zot/"
```

Résultat « brut » : personnalisation possible !

Reproductibilité ?

- Nettoyage de l’environnement
- Sauvegarde du graphe de dépendances concrétisé

  ⇒ fichier « spec.yaml » utilisable pour une réinstallation
- Conservation :
  - des fichiers recettes utilisés
  - de l’environnement de compilation
  - de la sortie de la compilation
  - d’autres fichiers importants (exemple : « config.log »)

Reproductibilité ?

- Des limites :
  - Utilisation de paquets externes
  - Oubli de dépendances dans les recettes
  - Dépendances systèmes bas niveau (glibc, ...)

Aperçu d'une recette

```python
class Vapor(CMakePackage):
  """VAPOR is the Visualization and Analysis Platform for Ocean, Atmosphere, and Solar Researchers."""

  homepage = "https://www.vapor.ucar.edu/"
  url = "https://github.com/NCAR/VAPOR/archive/3.3.0.tar.gz"

  version('3.3.0', sha256='508f93db9f6d9307be260820b878d054553aeb1719087a14770889f9e50a18ac')

  variant('gui', default=True, description='Build the GUI')

  depends_on('gl', when='+gui')
  depends_on('qt@5:+opengl+dbus', when='+gui')
  depends_on('python@3.6.0:3.6.99')
  # [...]

  def cmake_args(self):
    with open('site.local', 'w') as f:
      python = self.spec['python']
      f.write('set (PYTHONVERSION {0})\n'.format(python.version.up_to(2)))
      f.write('set (PYTHONDIR {0})\n'.format(python.home))
      f.write('set (PYTHONPATH {0})\n'.format(python.package.site_packages_dir))

    args = [
      '-DBUILD_OSP=OFF',
      self.define_from_variant('BUILD_GUI', 'gui')
    ]
    return args

  def setup_run_environment(self, env):
    env.set('VAPOR_HOME', self.prefix)
```

Métadonnées

Versions

Variantes

Dépendances

- Recettes écrites en Python
- Presque un DSL (héritages)
- Réutilisent le langage de spec

Contrôle de l'installation
Ici : paramètres CMake

Contrôle de l'environnement d'exécution

Spack à l'IDRIS

Avant Spack

- Installations effectuées manuellement :
  - login partagé
  - arborescences bien définies
- Procédures d'installation stockées sur un wiki
- Mélange au niveau des compilateurs
- Environment Modules 3

Gaussian install ada

Droits d'accès

L'IDRIS possède une site de site. Les utilisateurs doive

Le lien vers la procédure détaillée : [[1]]

Installation

Pire usine à gaz de tous les temps.

Passer en csh
Modifier les fichiers /bsd : set-mflags, setup-make,
setenv g09root PATH
source /bsd/g09.login
nohup /bsd/bldg09 >&make.log &

Sources à modifier :

getden.f
ou
denget.f

Une fonction écrase la constante IMeth
Le problème a été remonté à Gaussian

Étude de plusieurs outils

- À l'été 2018 en prévision du renouvellement de la machine
- Étude des fonctionnalités ⇒ Spack et EasyBuild retenus
  - Utilisent Python
  - Orientés HPC avec des utilisateurs en Europe
  - Compatibles avec Environment Modules

  ⇒ Ne pas trop perturber les utilisateurs et l'équipe
- Mise en pratique sur notre prototype Ouessant (Power8)

Résultats des tests

- Architecture exotique = recettes pas forcément adaptées
- EasyBuild :
  - Beaucoup de fichiers (toolchains, fichier « easyconfig »)
  - Perçu comme plutôt complexe à utiliser

- Spack : victoire par KO !
  - Enthousiasme pour le langage de spécifications
  - Perçu comme plus simple, même au niveau des recettes

- TensorFlow = échec dans les deux cas

Renforcement de la sécurité

- « Protection du potentiel scientifique et technique de la nation »
- IDRIS = Zone à Régime Restreint physique et virtuelle
- Installation de produits = administration, même sans root
- Login avec mot de passe partagé plus acceptable

Renforcement de la sécurité

- Nouvelles contraintes en vigueur depuis Jean Zay :
  - Machine dédiée aux installations
  - Système de fichiers monté avec accès minimaux
  - Passage par un bastion avec journalisation
  - Authentification à plusieurs facteurs

- Un peu d'inquiétude mais finalement transparent !

Spack en équipe ?

- Principale question : un login ou plusieurs ?
- Système de verrouillage perfectionné
  ⇒ pas de soucis avec les accès concurrents
- Problème : garantir l'accès en écriture à tous
  - Répertoires d'installation : plutôt bien gérés par Spack
  - Répertoires et fichiers annexes (ex : .pyc) : pas si sûr…

  ⇒ Choix mono-login compatible sécurité : « sudo -i -u »

Configuration de Spack

- Spack actuellement non accessible aux utilisateurs
- Respect de la philosophie de Spack
  ⇒ Limitation des paquets externes
- Paramétrages particuliers limités au maximum :
  - Structure de l'arborescence d'installation
  - Compilateur par défaut
  - Fournisseurs par défaut pour MPI, BLAS, LAPACK, etc

Mise à disposition des produits ?

- Utilisateurs habitués à Environment Modules
- Explosion combinatoire des modules !
- Étude de plusieurs solutions :
  - Modules classiques avec suffixes : netcdf/4.7.4-gcc-8.3.1
  - Approche hiérarchique : Lmod
    ⇒ charger un module en débloque d'autres
  - Approche pseudo-hiérarchique : Environment Modules 4
    ⇒ gestion automatique des dépendances

Environment Modules 4

- Pas d'évolution depuis 2012 (version 3.2.10)
- Développement repris par X. Delaruelle (CEA TGCC)
- Nombreux bugs corrigés
- Optimisations de performances
- Nouvelles fonctionnalités :
  - Nombreuses améliorations « ergonomiques »
  - **Gestion automatique des dépendances**
  - ...

Gestion automatique des dépendances

- Chargement automatique des prérequis

```
#%Module1.0

module-whatis "App"

prereq lib1
prereq lib2a lib2b
```

```
$ module load app
Loading app
  Loading requirement: lib1 lib2a

$ module unload app
Unloading app
  Unloading useless requirement: lib1 lib2a

$ module load lib2b
$ module load app
Loading app
  Loading requirement: lib1

$ module unload app
Unloading app
  Unloading useless requirement: lib1
```

Gestion automatique des dépendances

- Objectif à l'IDRIS : cohérence de l'environnement
- Prérequis :
  - Compilateur
  - Bibliothèque MPI
- Installation choisie en fonction de l'environnement chargé
- Chargement d'un environnement par défaut si vierge

Intégration avec Spack

- Modules générés pour Environment Modules 3
  ⇒ pas de support de l'approche pseudo-hiérarchique
- Développement d'une extension pour Spack :
  - Changement de modèle
    ⇒ « 1 module pour 1 installation » à 1 pour N
  - Code pas propre mais fonctionnel
  - Volonté d'upstreamer mais manque de temps...

Exemples

```
$ module help netcdf-c/4.7.4
-------------------------------------------------------------------
Module Specific Help for /.../netcdf-c/4.7.4:

NetCDF (network Common Data Form) is a set of software libraries and
machine-independent data formats that support the creation, access, and
sharing of array-oriented scientific data. This is the C distribution.

Available software environment(s):
- intel-compilers/19.1.2
- intel-compilers/19.1.1
- intel-compilers/19.0.4
- nvidia-compilers/20.7
- pgi/20.4

If you want to use this module with another software environment,
please contact the support team.
-------------------------------------------------------------------
```

Exemples

```
$ module help netcdf-c/4.7.4-mpi
-------------------------------------------------------------------
Module Specific Help for /.../netcdf-c/4.7.4-mpi:

NetCDF (network Common Data Form) is a set of software libraries and
machine-independent data formats that support the creation, access, and
sharing of array-oriented scientific data. This is the C distribution.

Available software environment(s):
- intel-compilers/19.1.2 intel-mpi/2019.8
- intel-compilers/19.1.1 intel-mpi/2019.7
- intel-compilers/19.1.1 openmpi/4.0.5
- intel-compilers/19.1.1 openmpi/4.0.4
- intel-compilers/19.0.4 intel-mpi/2019.4
- pgi/20.4 openmpi/4.0.4

If you want to use this module with another software environment,
please contact the support team.
-------------------------------------------------------------------
```

Exemples

```
$ module load netcdf-c/4.7.4
Loading netcdf-c/4.7.4
  Loading requirement: intel-compilers/19.1.2
$ which ncdump
/.../netcdf-c/4.7.4/intel-19.1.2-6qb4f4s4fszzyttlwbvvwui7iztx3bkg/bin/ncdump


$ module purge
$ module load nvidia-compilers/20.7
$ module load netcdf-c/4.7.4
$ which ncdump
/.../netcdf-c/4.7.4/nvhpc-20.7-5dwvxx727x7b62pcfatwpk7dy7vjsq57/bin/ncdump


$ module purge
$ module load netcdf-c/4.7.4-mpi
Loading netcdf-c/4.7.4-mpi
  Loading requirement: intel-compilers/19.1.2 intel-mpi/2019.8
$ which ncdump
/.../netcdf-c/4.7.4/intel-19.1.2-55jlam4hjbzwj2kh2rdcw4lcd2qpnpqq/bin/ncdump
```

Exemples

```
$ module purge
$ module load intel-compilers/19.1.1 intel-mpi/2019.7 netcdf-c/4.7.4-mpi
$ which ncdump
/.../netcdf-c/4.7.4/intel-19.1.1-kd7sjsicnhzgdmdbuvmol34szbhn7ytb/bin/ncdump


$ module purge
$ module load intel-compilers/19.1.1 openmpi/4.0.5 netcdf-c/4.7.4-mpi
$ which ncdump
/.../netcdf-c/4.7.4/intel-19.1.1-6bwguvlp26lyegv5lvjgnkmfg64lyzn7/bin/ncdump


$ module purge
$ module load intel-compilers/19.1.2 openmpi/4.0.5 netcdf-c/4.7.4-mpi
Loading intel-mpi/2019.8
  ERROR: intel-mpi/2019.8 cannot be loaded due to a conflict.
    HINT: Might try "module unload openmpi" first.

Loading netcdf-c/4.7.4-mpi
  ERROR: Load of requirement intel-mpi/2019.8 failed
```

Limitations

- Gestion des variantes :
  - Préfixes conservés pour MPI/CUDA
  - Approche alternative : « flavors »
- Gestion des compilateurs :
  - Pas d’exclusivité possible
  - Priorisation mais des problèmes potentiels

Cas des produits IA

- Maintenant plutôt bien gérés par Spack
- Utilisateurs de la communauté IA :
  - Fort attachement à Conda
  - Habitudes déjà bouleversées par les contraintes HPC

  ⇒ pas possible d'imposer Spack
- Approche 100% Conda impossible
  ⇒ Conda + Spack + installations manuelles

Conclusion

- Très gros progrès par rapport à l'existant
- Aucun regret sur le choix de Spack
- Contributions régulières de l'IDRIS pour les recettes
- Utilisateurs à priori satisfaits
- Impact visible sur les remontées de problèmes utilisateurs

  ⇒ Effet de la cohérence accrue de l'environnement

Fichiers de configuration

- Plusieurs répertoires avec des portées différentes :
  - Défauts : `<prefix>/etc/spack/defaults`
  - Système : `/etc/spack/`
  - Installation : `<prefix>/etc/spack/`
  - Personnel : `~/.spack`
- Fichiers au format YAML

Fichiers de configuration

- « config.yaml » : configuration générale

```
config:
  # This is the path to the root of the Spack install tree.
  # You can use $spack here to refer to the root of the spack instance.
  install_tree:
    root: $spack/opt/spack
    projections:
      all: "${ARCHITECTURE}/${COMPILERNAME}-${COMPILERVER}/${PACKAGE}-${VERSION}-${HASH}"

  # Temporary locations Spack can try to use for builds.
  build_stage:
  - $tempdir/$user/spack-stage

  # Cache directory for already downloaded source tarballs and archived
  # repositories. This can be purged with `spack clean --downloads`.
  source_cache: $spack/var/spack/cache

  # The maximum number of jobs to use when running `make` in parallel
  build_jobs: 8
```

Fichiers de configuration

- « compilers.yaml » : configuration des compilateurs

```
compilers:
- compiler:
  spec: gcc@7.5.0
  paths:
    cc: /usr/bin/gcc
    cxx: /usr/bin/g++
    f77:
    fc:
    flags: {}
  operating_system: ubuntu18.04
  target: x86_64
  modules: []
  environment: {}
  extra_rpaths: []
```