INSTITUT DU DÉVELOPPEMENT ET DES RESSOURCES EN INFORMATIQUE SCIENTIFIQUE

www.idris.fr

Groupe Calcul
Atelier gestionnaires de paquets
27/11/2025

Introduction à Spack

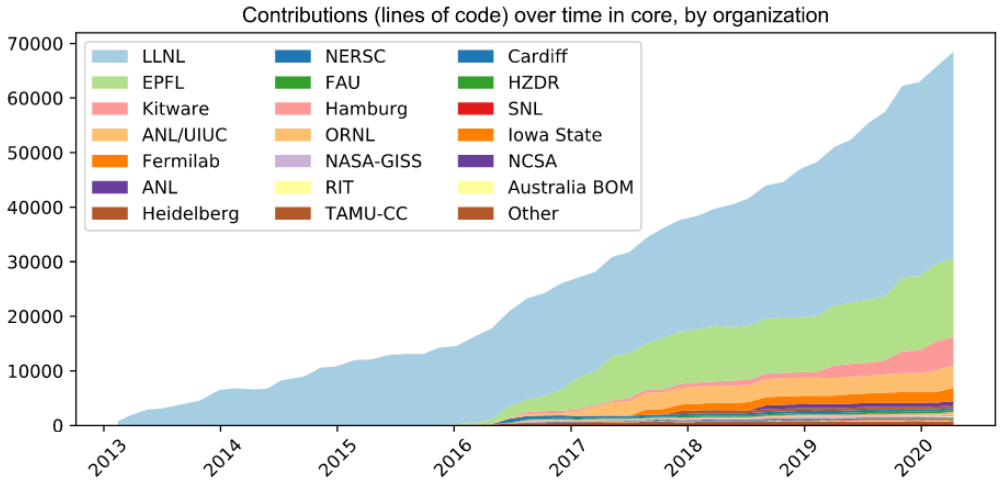
Rémi Lacroix

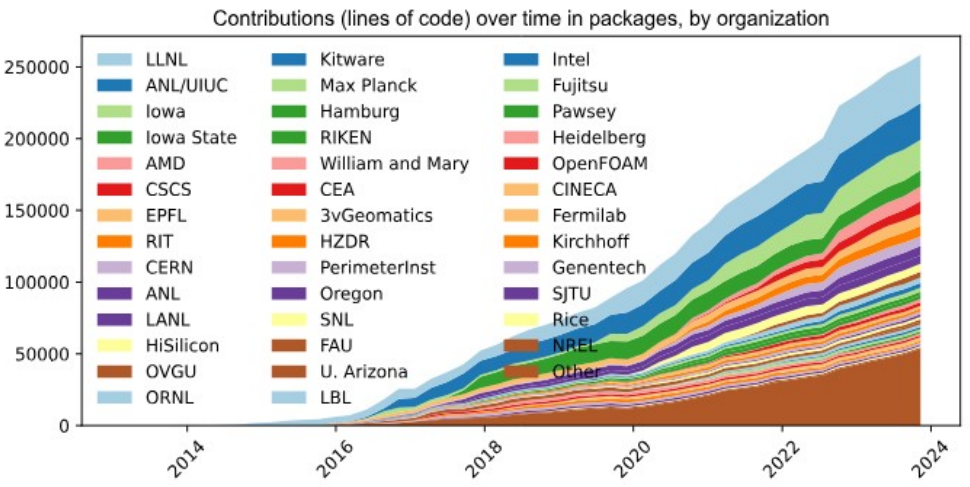
Spack

- **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 :



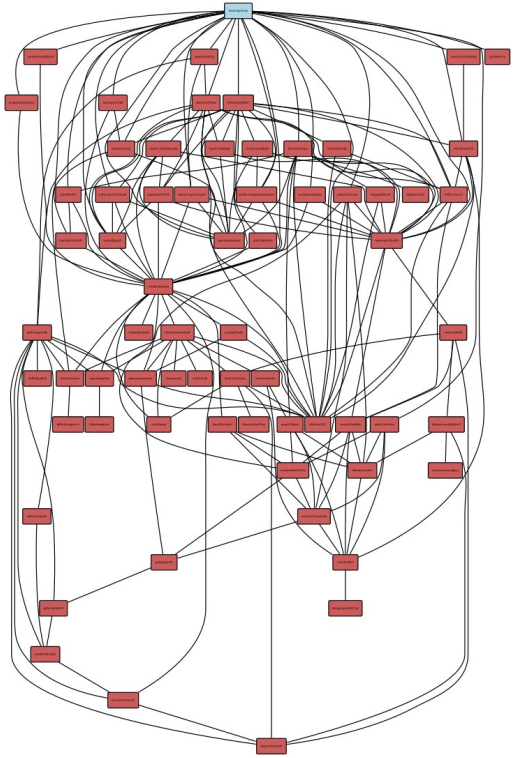


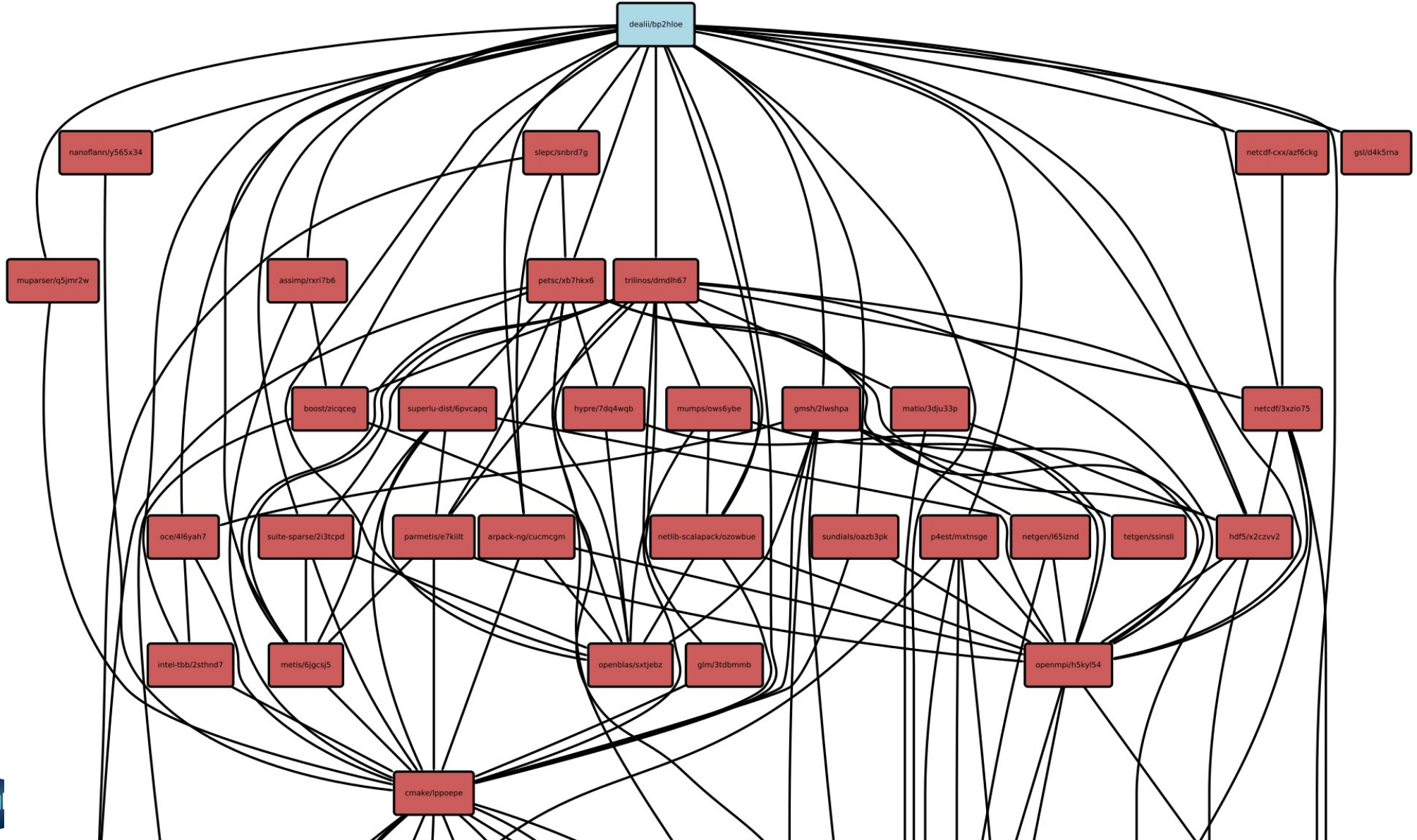
Un projet actif

- Bonne implantation :
  - High Performance Software Foundation / Linux Fondation
  - 3 premières machines au TOP500 :
    - El Capitan (LLNL, USA)
    - Frontier (ORNL, USA)
    - Aurora (ANL, USA)

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, NVIDIA, 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 --depth=2 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 à la première installation

```
$ spack compilers
==> Available compilers
-- gcc ubuntu22.04-x86_64 ---------------------------------
[e]  gcc@9.5.0  [e]  gcc@11.4.0
```

- Ajoutés à la demande

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

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

- Installés avec Spack

Paquets supportés

- De plus en plus nombreux !

```
$ spack list
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
==> 8621 packages
```

- Filtrage possible

```
$ spack list netcdf
netcdf-c  netcdf-cxx  netcdf-cxx4 ...  netcdf-fortran  parallel-netcdf  py-netcdf4
==> 9 packages

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

Détails d'un paquet

```
$ spack info netcdf-c
CMakePackage:   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

Preferred version:
    4.9.3      https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.9.3.tar.gz

Safe versions:
    main       [git] https://github.com/Unidata/netcdf-c.git on branch main
    4.9.3      https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.9.3.tar.gz
    4.9.2      https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.9.2.tar.gz
...
    4.3.3      https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.3.3.tar.gz

Deprecated versions:
   None

[A suivre à la prochaine diapositive]
```

Versions connues par Spack

Détails d'un paquet

```
$ spack info netcdf-c [Suite et fin]
Variants:
    blosc [true]                    false, true
        Enable Blosc compression plugin

    build_system [autotools]        autotools, cmake
        Build systems supported by the package

    build_type [Release]            Debug, MinSizeRel, RelWithDebInfo, Release
      when  build_system=cmake
        CMake build type

...

Dependencies:
    autoconf                        build
      when @main build_system=autotools
...
    c                               build
...
    hdf5@:1.10                      build, link
      when @:4.7.3
...

Licenses:
    BSD-3-Clause
```

Les options

Les différentes dépendances

Base de données

- Conserve la trace de toutes les installations

```
$ spack find
-- linux-ubuntu22.04-skylake_avx512 / %c,cxx=gcc@9.5.0 ----------
berkeley-db@18.1.40  gettext@0.23.1  m4@1.4.20            texinfo@7.2    zstd@1.5.7
gcc@14.3.0           gmp@6.3.0       ncurses@6.5-20250705  zlib-ng@2.2.4
...

-- linux-ubuntu22.04-skylake_avx512 / no compilers --------------
autoconf@2.72  autoconf-archive@2023.02.20  compiler-wrapper@1.0  gcc-runtime@9.5.0
...

==> 34 installed packages
```

- 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 / %c=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
...
```

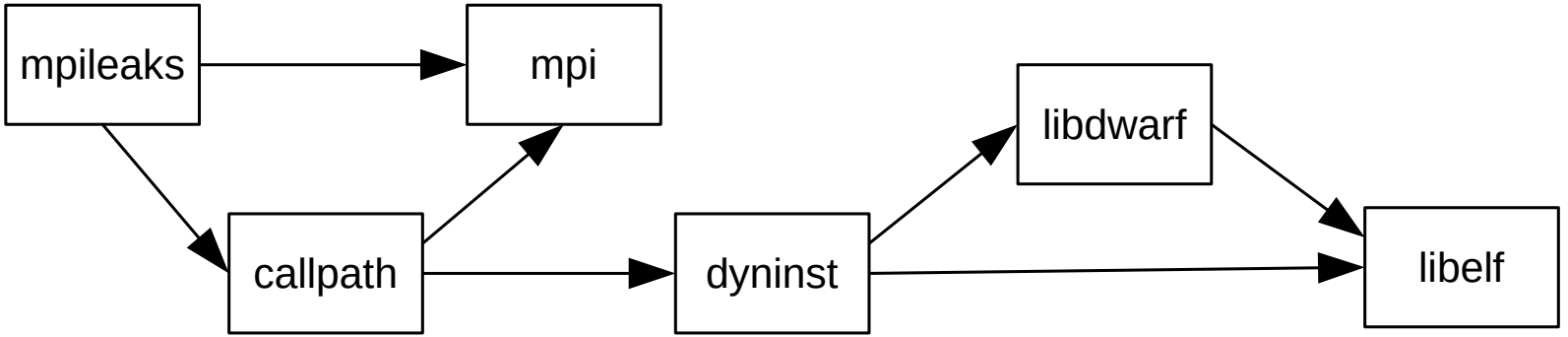
Langage de spécification

- 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 / %c=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

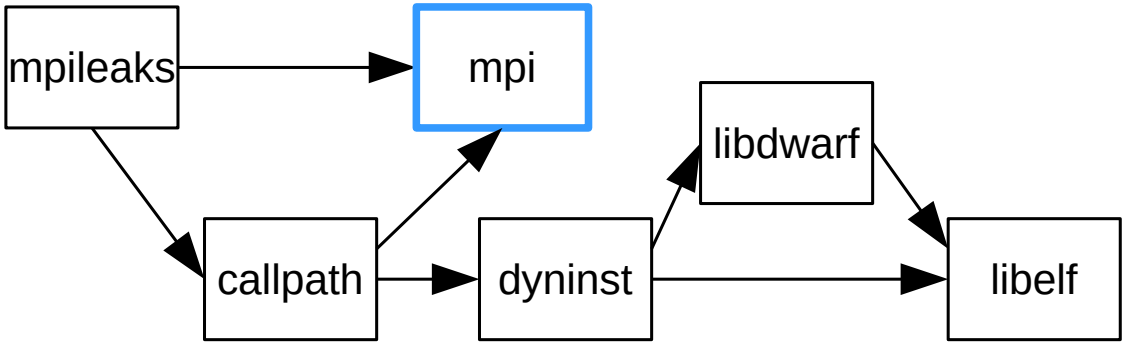
```
$ spack find -Lv /x6rxxa
...
-- linux-ubuntu18.04-broadwell / %c=gcc@7.5.0 --------------------
x6rxxaqcl3lgzvkl7qwzc6pkkmxfjdkh zlib@1.2.11+optimize+pic+shared

$ 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
```

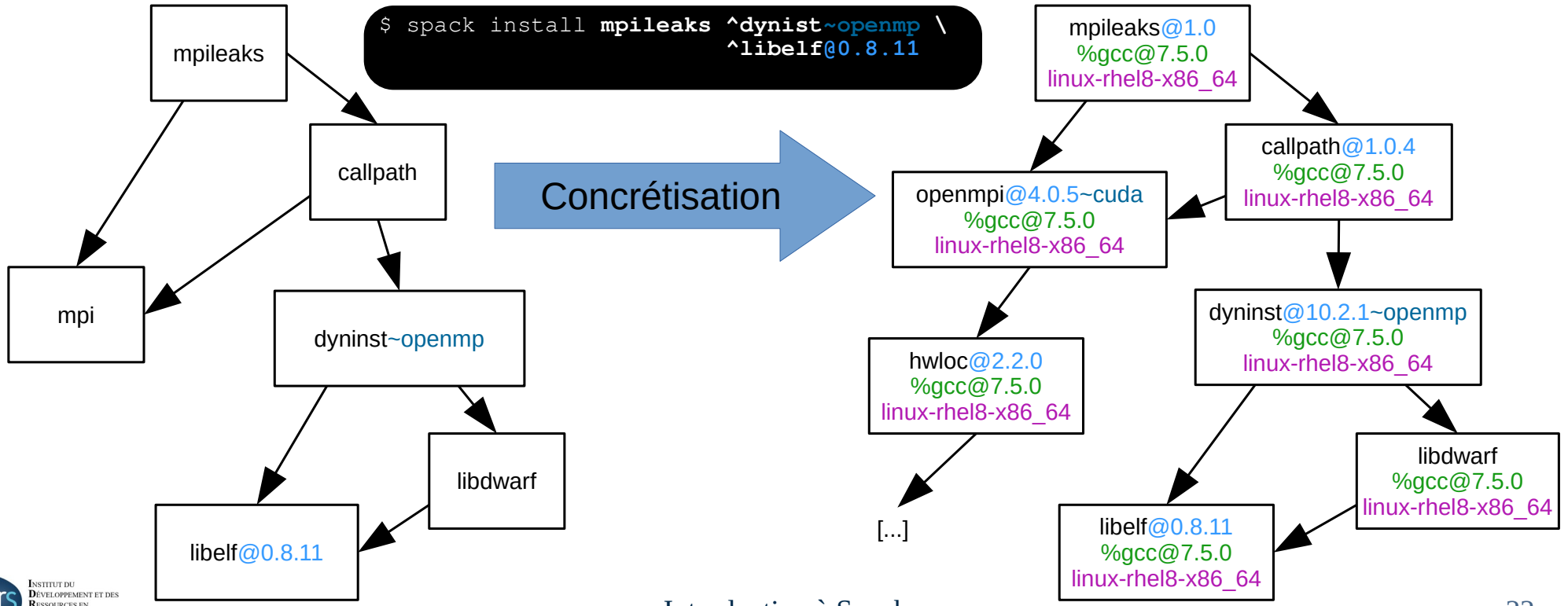
Dépendances virtuelles

- Possibilité de lister les « fournisseurs » d'une dépendance virtuelle

```
$ spack providers mpi@3
-- no arch / no compilers ---------------------------------------
cray-mpich      intel-oneapi-mpi  mpich          mpt@3:         mvapich2       nvhpc
cray-mvapich2   mpi-serial        mpilander      msmpi          mvapich2@2.1:  openmpi@1.7.5:1.10.7
fujitsu-mpi     mpich@:3.1        mpitrampoline  mvapich        mvapich2@2.3:  openmpi@2.0.0:
hpcx-mpi        mpich@:3.2        mpt            mvapich-plus   nmad           spectrum-mpi
```

Concrétisation

- Complète la spécification abstraite fournie par l'utilisateur

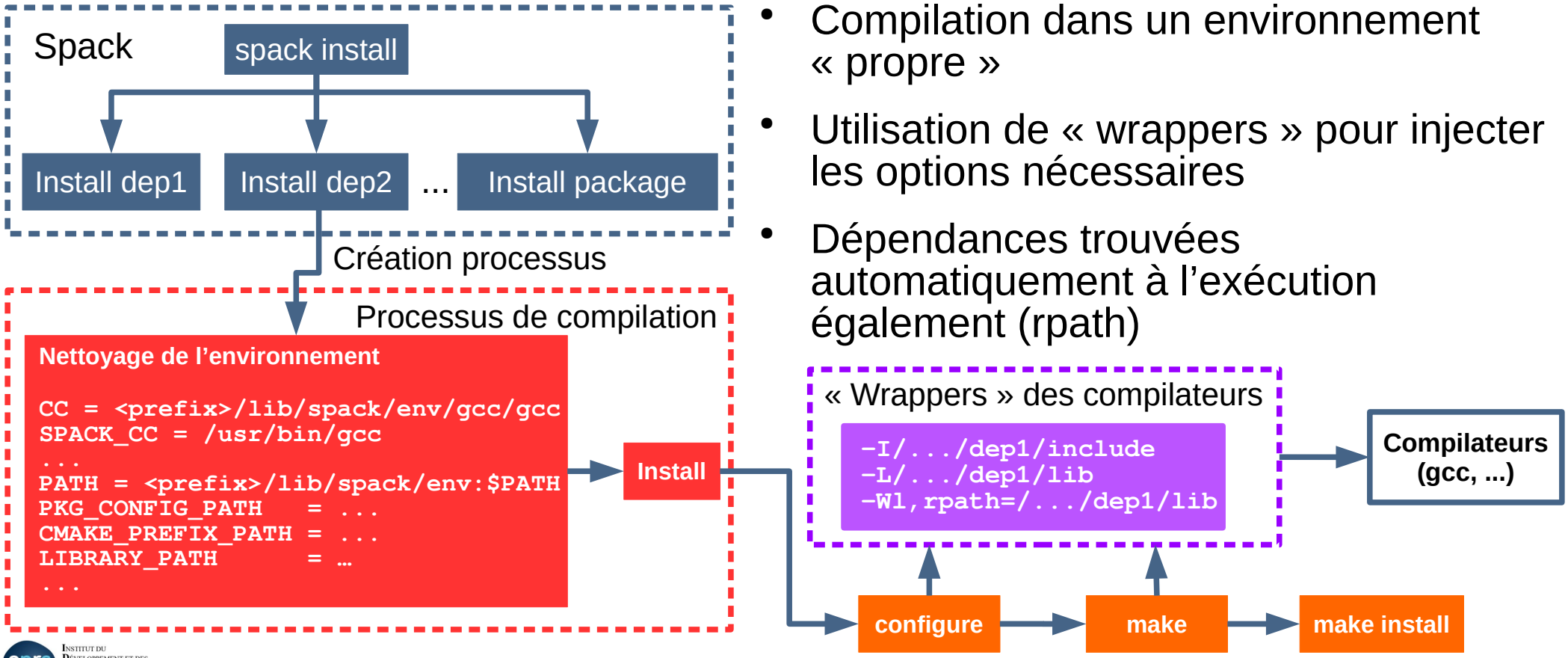


Concrétisation

- Possibilité de la visualiser avant l'installation

```
$ spack spec -It netcdf-c~mpi ^hdf5~mpi+szip
-   [    ]  netcdf-c@4.9.3~blosc~byterange~dap~fsync~hdf4~jna~logging~mpi~nczarr_zip+optimize~parallel-
netcdf+pic+shared+szip+zstd build_system=autotools platform=linux os=ubuntu22.04 target=skylake_avx512
%c=gcc@9.5.0
[+]  [bl  ]      ^bzip2@1.0.8~debug~pic+shared build_system=generic platform=linux os=ubuntu22.04
target=skylake_avx512 %c=gcc@9.5.0
[+]  [b   ]          ^diffutils@3.12 build_system=autotools platform=linux os=ubuntu22.04
target=skylake_avx512 %c=gcc@9.5.0
[+]  [bl  ]              ^libiconv@1.18 build_system=autotools libs:=shared,static platform=linux
os=ubuntu22.04 target=skylake_avx512 %c=gcc@9.5.0
[+]  [b   ]      ^compiler-wrapper@1.0 build_system=generic platform=linux os=ubuntu22.04
target=skylake_avx512
[e]  [b   ]      ^gcc@9.5.0~binutils+bootstrap~graphite~nvptx~piclibs~profiled~strip
build_system=autotools build_type=RelWithDebInfo languages:='c,c++' platform=linux os=ubuntu22.04
target=x86_64
[+]  [ l  ]      ^gcc-runtime@9.5.0 build_system=generic platform=linux os=ubuntu22.04
target=skylake_avx512
[e]  [ l  ]      ^glibc@2.35 build_system=autotools platform=linux os=ubuntu22.04 target=x86_64
[+]  [b r ]      ^gmake@4.4.1~guile build_system=generic platform=linux os=ubuntu22.04
target=skylake_avx512 %c=gcc@9.5.0
-   [bl  ]      ^hdf5@1.14.6~cxx~fortran+hl~ipo~java~map~mpi+shared+szip~threadsafe+tools api=default
build_system=cmake build_type=Release generator=make platform=linux os=ubuntu22.04 target=skylake_avx512
%c=gcc@9.5.0
-   [b   ]          ^cmake@3.31.9~doc+ncurses+ownlibs~qtgui build_system=generic build_type=Release
platform=linux os=ubuntu22.04 target=skylake_avx512 %c,cxx=gcc@9.5.0
...
```

Mécanisme de compilation



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

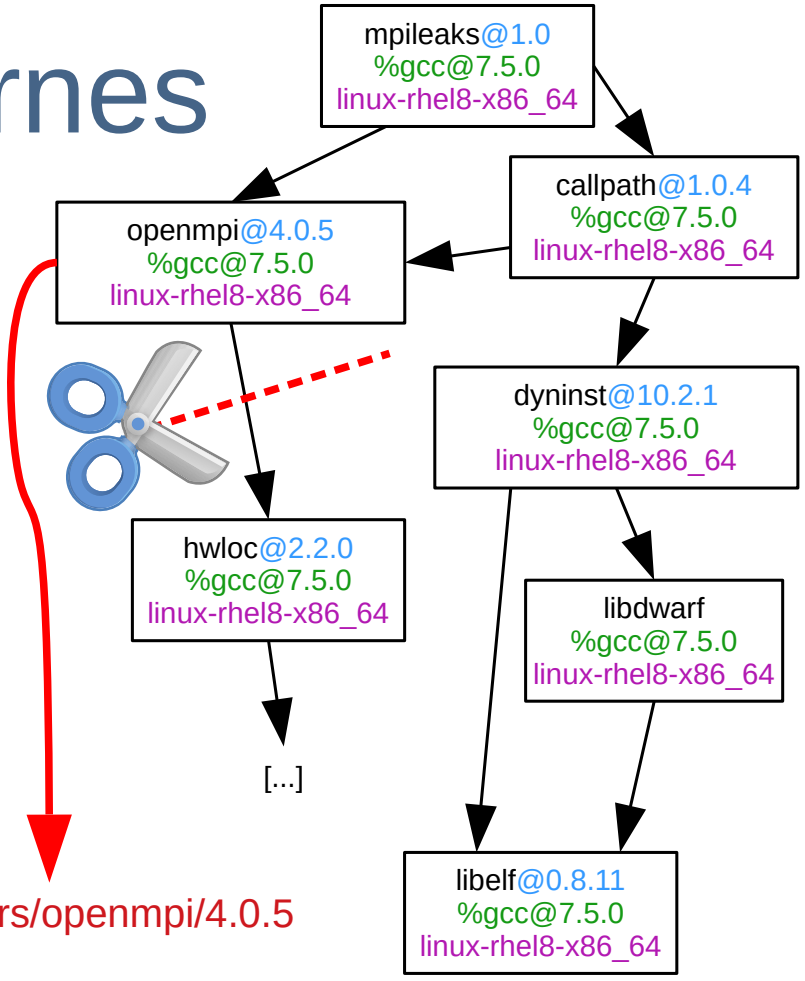
Paquets externes

- « 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

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}/${compiler.name}-${compiler.version}/${name}-${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

- « packages.yaml » : configuration des installations externes, y compris les compilateurs

```
packages:
  gcc:
    externals:
    - spec: gcc@9.5.0 languages:='c,c++'
      prefix: /usr
      extra_attributes: /usr
        compilers:
          c: /usr/bin/gcc-9
          cxx: /usr/bin/g++-9
    - spec: gcc@11.4.0 languages:='c,c++'
      prefix: /usr
      extra_attributes:
        compilers:
          c: /usr/bin/gcc
          cxx: /usr/bin/g++
```