

Spack for Beginners

Karim Hasnaoui

Institut du Développement et des Ressources en Informatique Scientifique, UAR851 – Orsay
Maison de la Simulation UAR3441 - Saclay

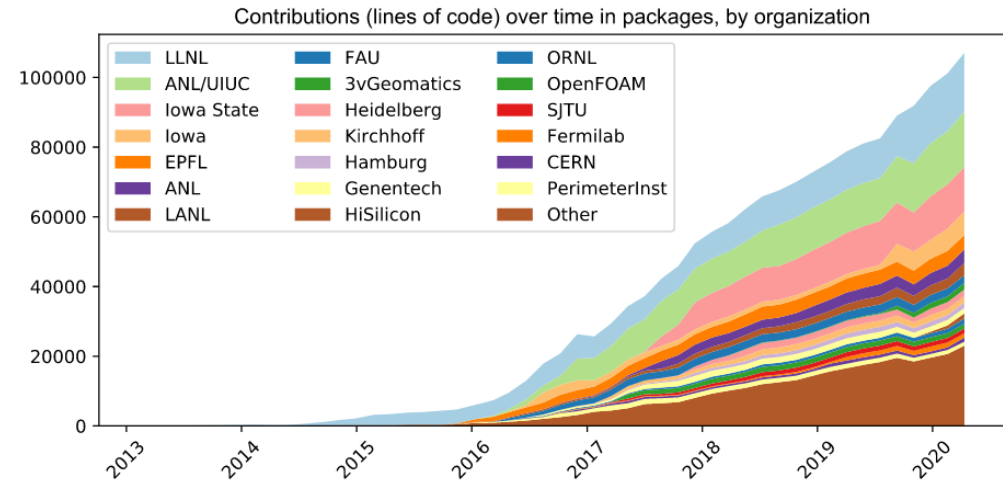
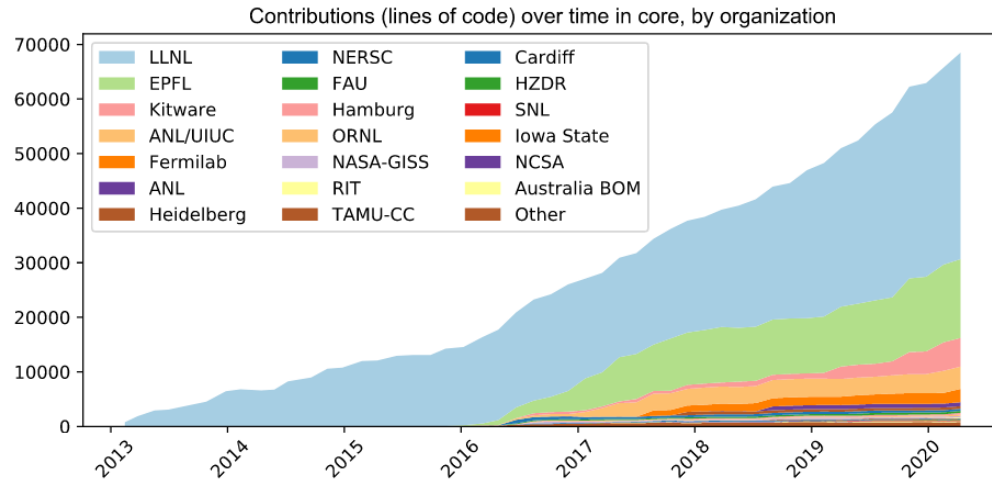
Based on Rémi Lacroix's talk at ANF and the Spack Tutorial (<https://spack-tutorial.readthedocs.io/>) distributed under the MIT License Copyright (c) 2013-2025 LLNS, LLC and other Spack Project Developers.



- Supercomputer **PACK**age manager
- Source-based package manager
- Developed at Lawrence Livermore National Laboratory since 2013
- Originally inspired by Homebrew and Nix
- Focused on scientific computing and HPC

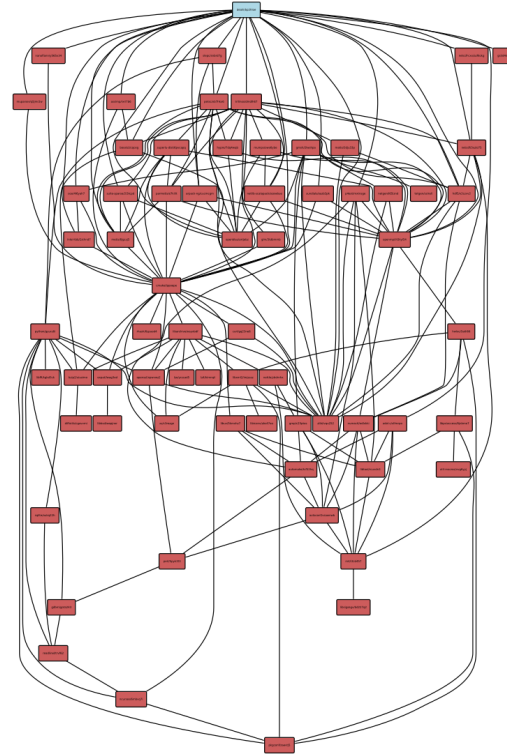
An actively maintained project

- Numerous contributions from a wide range of sources



Dependency hell

- Example of the Deal.II library (finite elements) :



Combinatorial explosion

- Real-life example from Jean Zay supercomputer :
 - Dozens of products and libraries
 - Three compilers: GNU, Intel, and NVHCP
 - Two MPI implementations: OpenMPI and Intel MPI
 - Multiple versions of each...
 - OpenMP/OpenACC/CUDA

Goals of Spack

- Simplify dependency management
- Allow easy changes to :
 - versions
 - compiler and its options
 - libraries such as MPI, BLAS, LAPACK, ...
- Improve reproducibility of installations without compromising performance

Installing Spack

- Cloning the Git repository is sufficient

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

- With a compiler installed :

```
$ spack install zlib
```

- By default, installation in the « opt » subdirectory"

Compilers

- Automatically detected on first launch

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

- Added on demand

```
$ spack compiler add          # Cherche les compilateurs dans le PATH
$ spack compiler add /rep/install/compilo # Recherche avec un indice
```

- Configuration file : compilers.yaml

Supported software

- Growing daily!

```
$ 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
```

- Filtering available

```
$ 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
```

Package details

```
$ 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]
```

Known versions

Package details

```
$ spack info netcdf-c
```

```
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

Options

```
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
```

Different
dependencies

Database

- Keeps track of all 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
```

- Ability
 - to support multiple installations of the same product
 - to manage different architectures

Database

- Dependency tree

```
$ 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
...

```

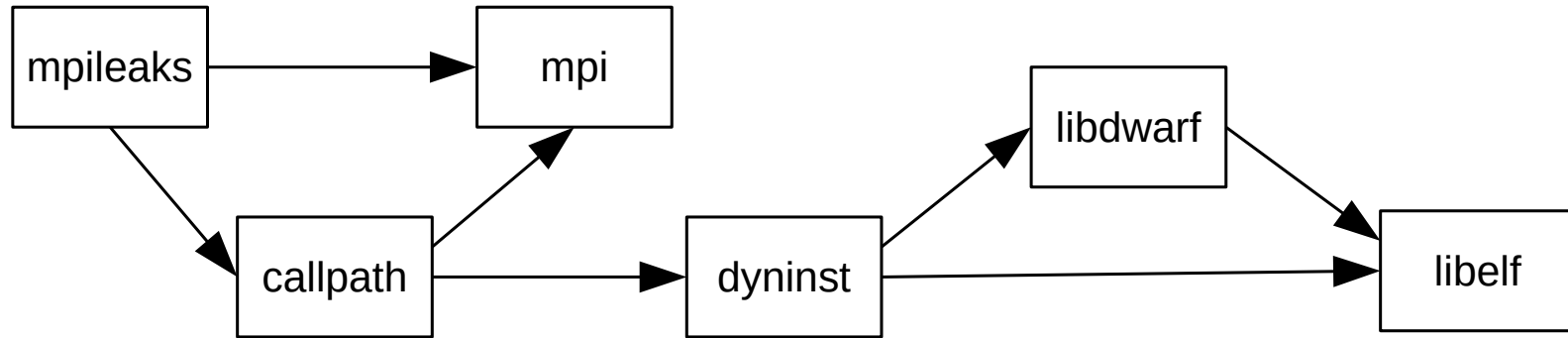
Specification language

- Spack's greatest strength

```
$ spack install netcdf-c # No constraint
$ spack install netcdf-c@4.7.3 # Specific version
$ spack install netcdf-c%gcc@8.3.1 # Specific compiler
$ spack install netcdf-c+mpi~hdf4 # Specific variants
$ spack install netcdf-c ^openmpi # Specific dependency
$ spack install netcdf-c cflags="-O3 -g" ldflags="-O3 -g" # Specific compilation options/flags
$ spack install netcdf-c@4.7.3%gcc@8.3.1 ^hdf5@10.1.7+h1 # Recursive combination of
# all possibilities !
```

Dependency graph

- One installation = one directed acyclic graph



- Guaranteed consistency :
 - No mixing of versions in dependencies
 - Compiler mixing allowed with caution

Dependency graph

- A dependency graph = a unique fingerprint
 - ⇒ Everything is taken into account (versions, variants, etc)
- Fingerprint used in installation directories
 - ⇒ Solution to combinatorial explosion

```
opt
├── spack
│   ├── linux-ubuntu18.04-broadwell
│   │   ├── gcc-7.5.0
│   │   │   ├── hdf5-1.10.7-5umosquucqcsgeq6l56bpyeze5lr5y5l
│   │   │   ├── hdf5-1.10.7-1c2vgm75g5avg3enp6pi6wygrkdkmoe4
│   │   │   ├── libsigsegv-2.12-cy6vvhkposdmsbcu6vziegcxyxnlrmir
│   │   │   ├── libzip-2.1.1-m7btczwncfg4xkgvh55itdrqea3om22o
│   │   │   ├── m4-1.4.18-ddc3vhixyli3j7x32iovn2hrbqhfm5g5
│   │   │   ├── netcdf-c-4.7.4-az3aojtywmb5yntsx6rotgdgnqf55zot
│   │   │   └── zlib-1.2.11-x6rxxaqcl3lgzvkl7qwzc6pkkmxfjdkh
```


Fingerprint and database"

- Integral part of the database

```
$ spack find -lv hdf5
...
-- linux-ubuntu18.04-broadwell / gcc@7.5.0 -----
1c2vgm7 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
```

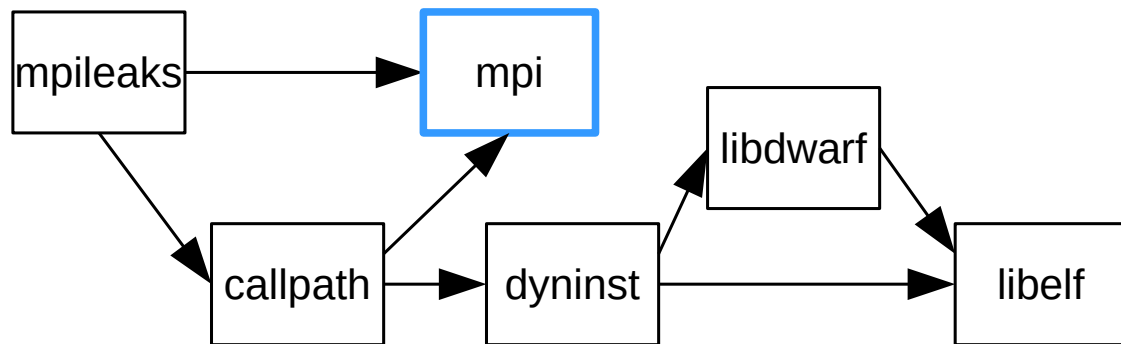
- Reference to a fingerprint

```
$ spack find -Lv /x6rxxa
...
-- linux-ubuntu18.04-broadwell / gcc@7.5.0 -----
x6rxxaqcl3lgzvk17qwzc6pkkmxfjdkh zlib@1.2.11+optimize+pic+shared

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

Virtual dependencies

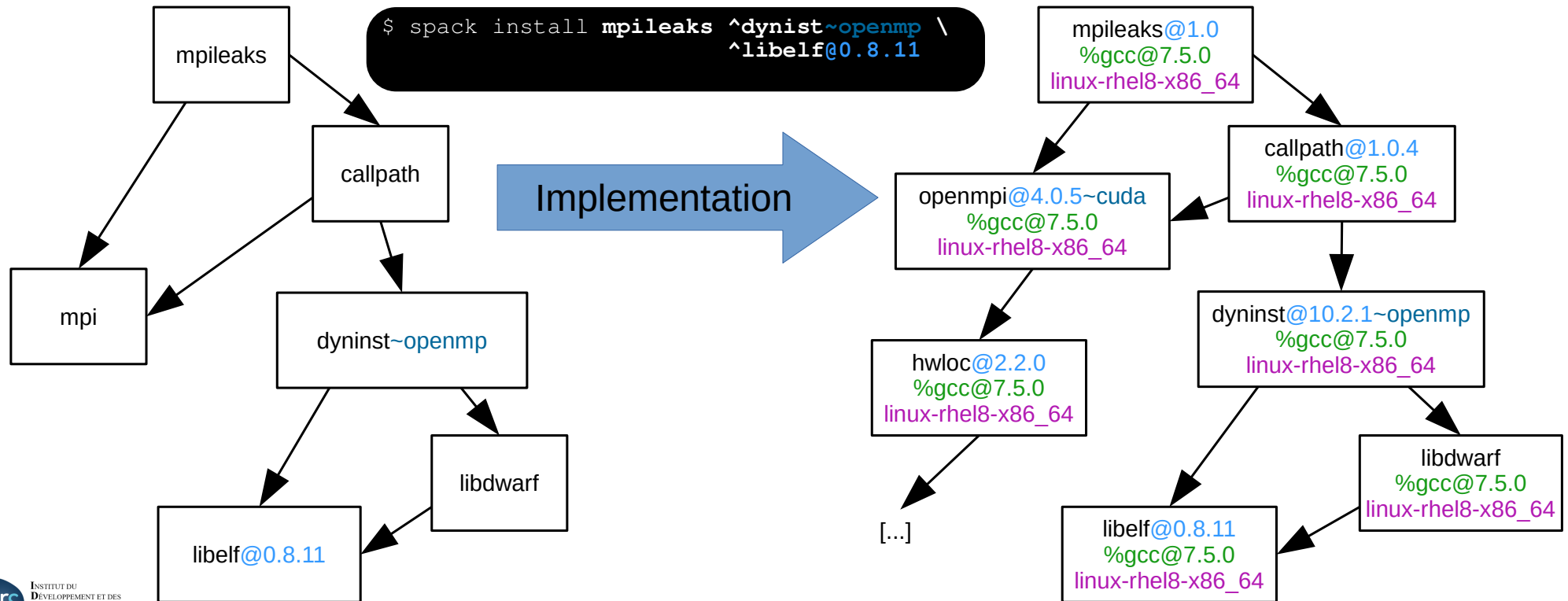
- Libraries with :
 - same API (potentially versioned)
 - but not the same ABI
- ⇒ Elegant solution for managing MPI, LAPACK, etc...



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

Concrete implementation

- Completes the abstract specification provided by the user



Concrete implementation

- Option to visualize it before installation

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

```
Input spec
```

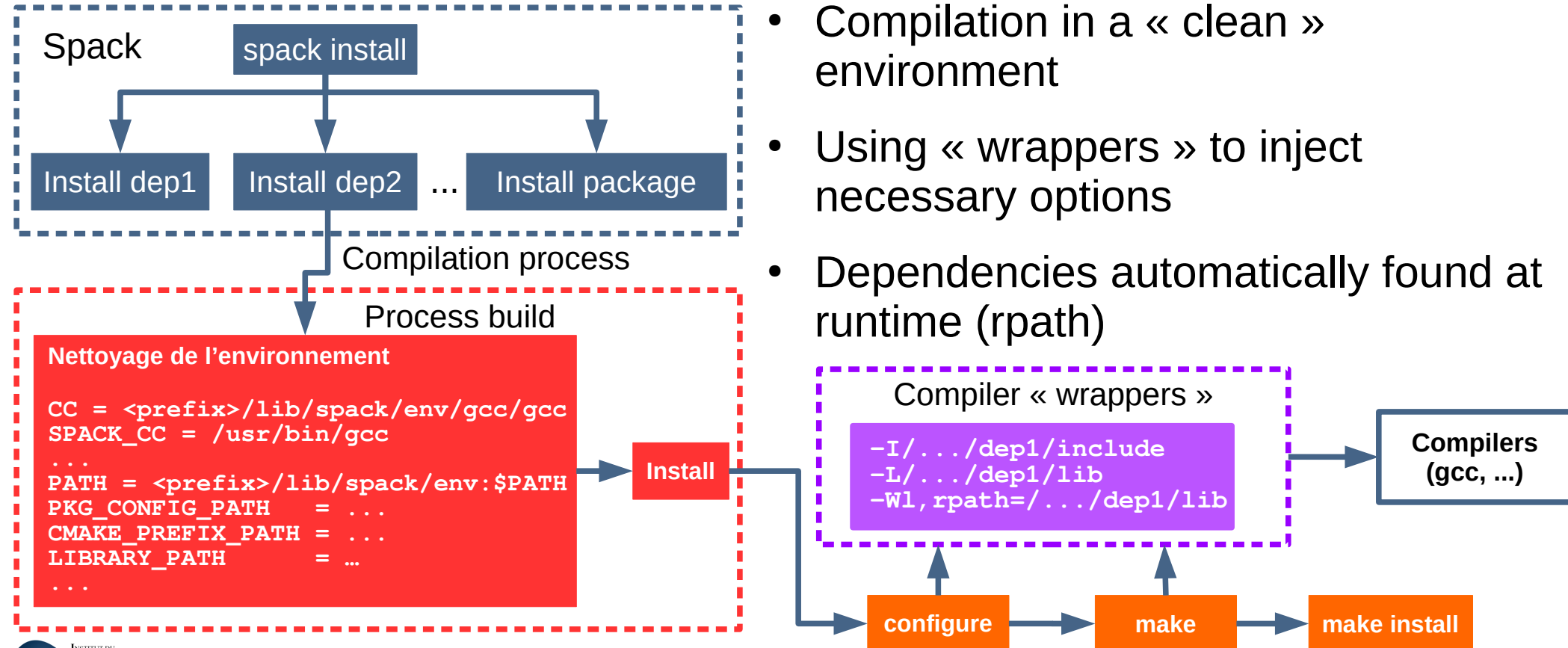
```
-----  
- [ ] netcdf-c~mpi  
- [ ] ^hdf5~mpi+szip
```

```
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
```

Compilation mechanism

- Compilation in a « clean » environment
- Using « wrappers » to inject necessary options
- Dependencies automatically found at runtime (rpath)



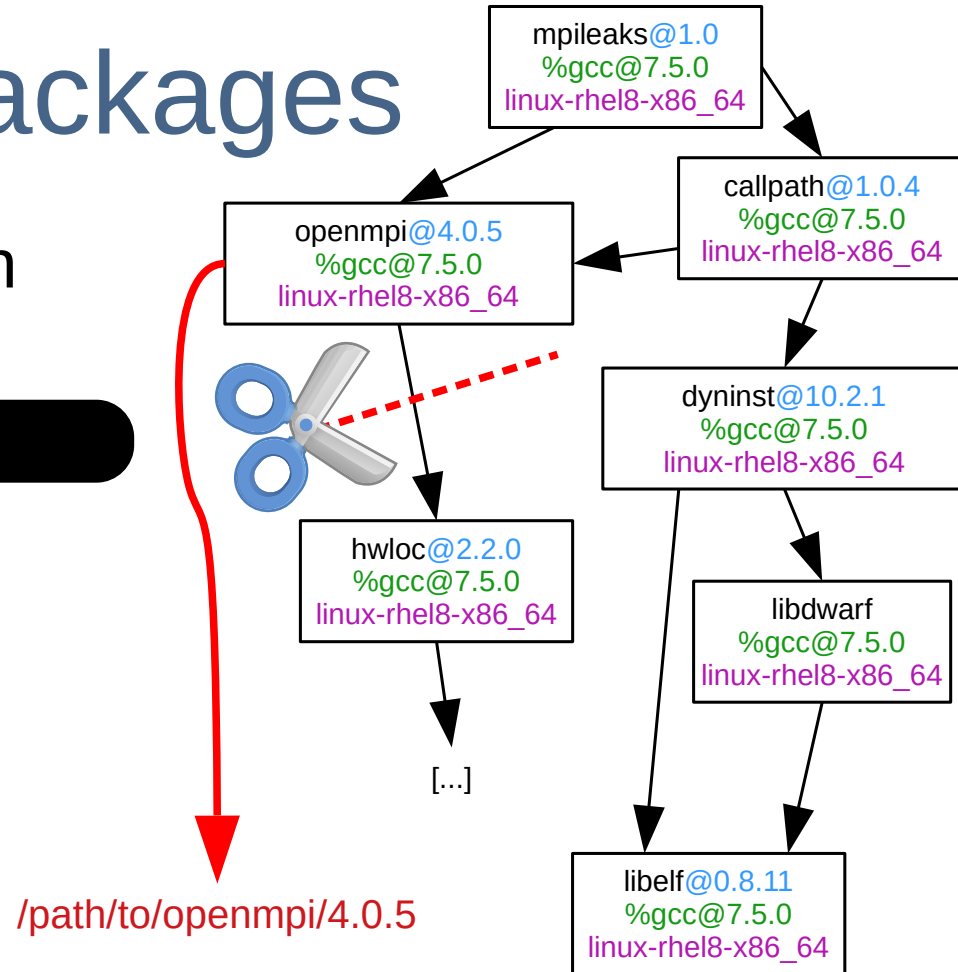
External packages

- « Cut » the graph to point to an external resource

```
$ 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
```



/path/to/openmpi/4.0.5

Optimizations

- Architecture detection

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

- Conversion to compilation options/flags
Ex : `-march=skylake-avx512 -mtune=skylake-avx512`
- Injected via compiler « wrappers »

Using installed products

- Directly through Spack :

```
$ which ncdump
```

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

```
$ which ncdump
```

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


Using installed products

- Via modules generated automatically by 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/"
```

Raw result :
Possible customization !"

Reproducibility ?

- Environment cleanup
- Saving the dependency graph
 - ⇒ « spec.yaml » file for reinstallation
- Saving :
 - used recipe files
 - compilation environment
 - compilation output
 - other important files (e.g. « config.log »)

Reproducibility ?

- Limitations :
 - Use of external packages
 - Missing dependencies in recipes
 - Low-level system dependencies (glibc, ...)

```

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)

```

Metadata

Versions

Variants

Dependencies

- Recipes written in Python
- DSL-like (with inheritance)
- Reuse the spec language

Installation control
Here: CMake parameters

Runtime environment control

Configuration files

- Multiple directories with different scopes :
 - Defaults directory : `<prefix>/etc/spack/defaults`
 - System directory : `/etc/spack/`
 - Installation directory : `<prefix>/etc/spack/`
 - User : `~/ .spack`
- YAML files

Configuration files

- « config.yaml » : General configuration

```
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
```

Configuration files

- « compilers.yaml » : compiler configurations

```
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: []
```