Build [Optimized] Images with Spack
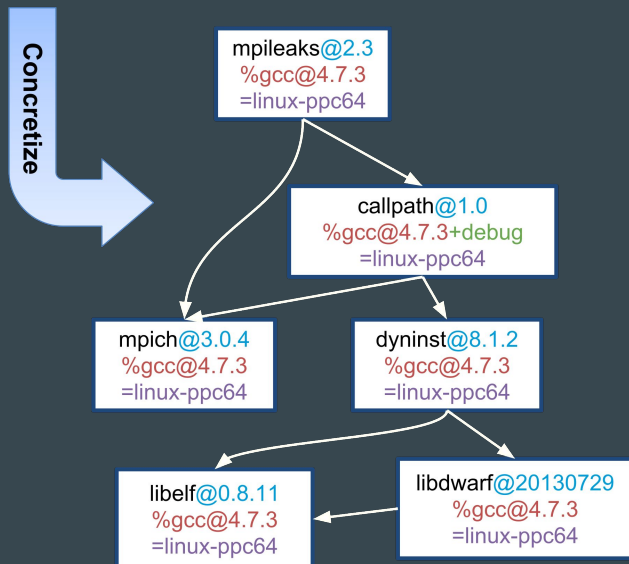
●●●

Massimiliano Culpo, EPFL

Spack manages complex application DAGs

mpileaks ^callpath@1.0 +debug ^libelf@0.8.11

Concretize

```
mpileaks@2.3
%gcc@4.7.3
=linux-ppc64
```

```
callpath@1.0
%gcc@4.7.3+debug
=linux-ppc64
```

```
mpich@3.0.4
%gcc@4.7.3
=linux-ppc64
```

```
dyninst@8.1.2
%gcc@4.7.3
=linux-ppc64
```

```
libelf@0.8.11
%gcc@4.7.3
=linux-ppc64
```

```
libdwarf@20130729
%gcc@4.7.3
=linux-ppc64
```

```
# Install a specific version by appending @
$ spack install hdf5@1.10.1

# Specify a compiler (and optional version), with %
$ spack install hdf5@1.10.1 %gcc@4.7.3

# Add special boolean compile-time options with +
$ spack install hdf5@1.10.1 +szip

# Add custom compiler flags
$ spack install hdf5@1.10.1 cppflags="-O3
-floop-block"

# Cross-compile on a Cray or Blue Gene/Q
$ spack install hdf5@1.10.1 target=backend

# Recurse to dependency with ^
$ spack install hdf5 ^zlib@1.2.8
```
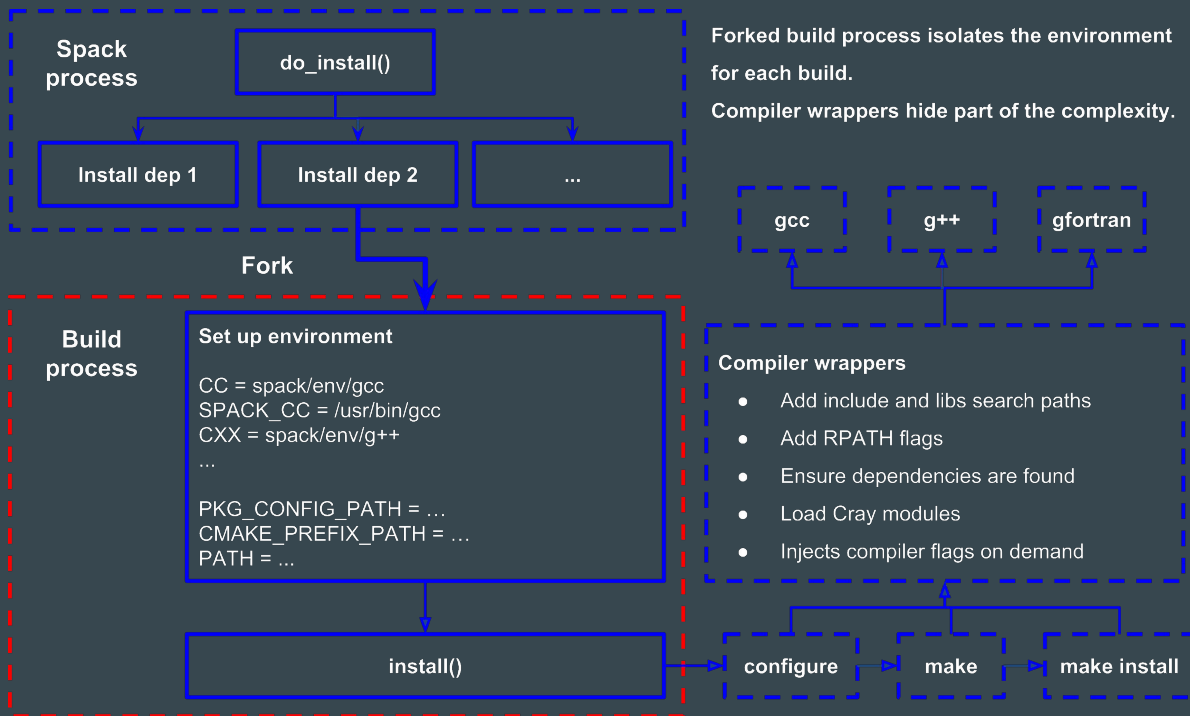
Compiler flags can already be injected programmatically

Spack process

do_install()

Install dep 1   Install dep 2   ...

Fork

Build process

Set up environment

CC = spack/env/gcc
SPACK_CC = /usr/bin/gcc
CXX = spack/env/g++
...

PKG_CONFIG_PATH = …
CMAKE_PREFIX_PATH = …
PATH = …

install()

Forked build process isolates the environment for each build.

Compiler wrappers hide part of the complexity.

gcc   g++   gfortran

Compiler wrappers
- Add include and libs search paths
- Add RPATH flags
- Ensure dependencies are found
- Load Cray modules
- Injects compiler flags on demand

configure   make   make install

Target support will be extended in the near future

```
# Compile for skylake. This will automatically
# inject flags like -march=skylake -mtune=skylake
$ spack install hdf5@1.10.1 target=skylake
```

Improvements on target support in the near future include:

- Better auto-detection of host micro-architecture
- Mapping micro-architecture and compilers to a proper set of optimization flags

The <u>long term goal</u> is to support software cross compilation, even across architectures.

https://github.com/spack/spack/pull/3206

Entire environments can be described in YAML format

```
# This is a Spack Environment file.
#
# It describes a set of packages to
# be installed, along with configuration
# settings.
spack:
  # add package specs to the `specs` list
  specs:
  -gromacs target=skylake
  -cp2k    target=skylake
```

Spack environments are built on the *manifest + lockfile* concept pioneered by Bundler.

We have experimental Spack containers: no need to replicate the set-up from base images to have Spack.

```
Bootstrap: docker
From: ubuntu:18.04

%files
$PWD/spack.yaml <workdir>/spack.yaml    ⬅
...

%post
# Install basic system dependencies
apt-get update && ...
# Install spack + set the mirror
mkdir -p $SPACK_ROOT
curl -s -L <spack-url> \
    | tar xzC $SPACK_ROOT --strip 1

cd <workdir>
/home/spack/bin/spack install    ⬅

%runscript
...
```

Features and improvements that are being discussed

- Closer integration with different container technology:
  - https://github.com/spack/spack/pull/7204 (docker)
  - https://github.com/spack/spack/pull/10952 (hpc-container-maker)
  - https://github.com/spack/spack/pull/11367 (singularity)
- Strip binaries to reduce container size
- Support for multi-stage builds (build deps pruned from the final image)
- Optionally unroll the DAG to make use of the container cache where available
- ...