Spack Overview and State of the Project

HPC U. Ghent

Todd Gamblin
Center for Applied Scientific Computing, LLNL

February 2, 2018

LLNL-PRES-745858

Goals

- Facilitate experimenting with performance options.
- Flexibility.

- Make these things easy:
  - Build the software in arbitrarily many different configurations
  - Swapping a new compiler into a build
  - Swapping implementations of libraries
    - e.g., MPI, BLAS, LAPACK, others like jpeg/jpeg-turbo, etc.
  - Injecting compiler flags into builds

- Build all the things you need for scientific analysis
- Run on laptops, Linux clusters, and the largest supercomputers

Spack

A flexible package manager for HPC

github.com/spack

@spackpm

Easy installation

```
$ git clone https://github.com/spack/spack
$ . spack/share/spack/setup-env.sh
$ spack install hdf5
```

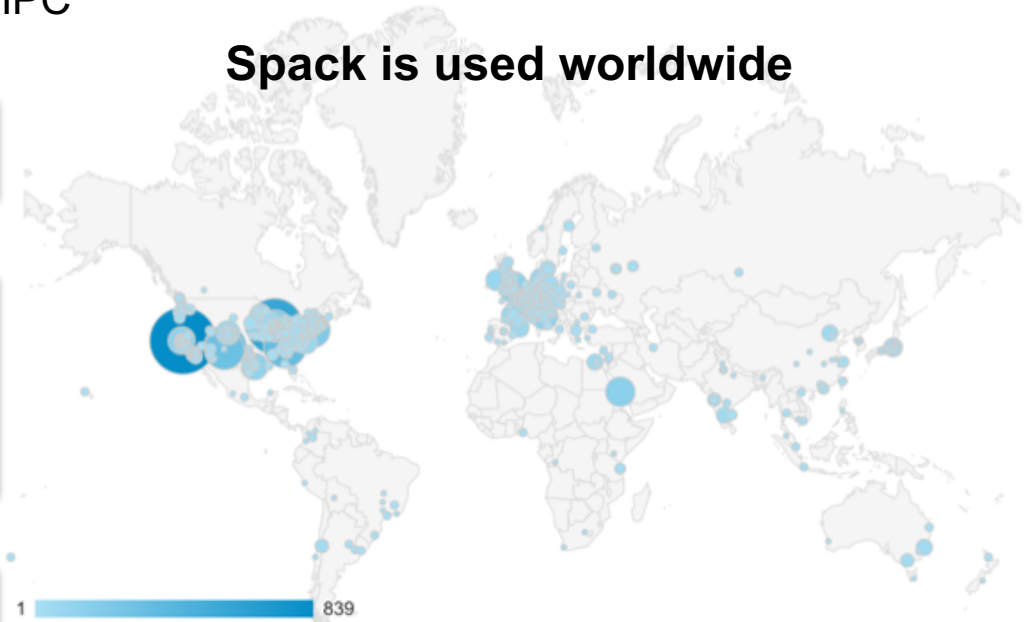
Easily experiment with build options

```
$ spack install mpileaks@3.3
$ spack install mpileaks@3.3 %gcc@4.7.3 +threads
$ spack install mpileaks@3.3 cppflags="-O3 -g3"
$ spack install mpileaks@3.3 target=haswell
$ spack install mpileaks@3.3 ^mpich@3.2
```

Easily share package recipes

```
from spack import *

class Kripke(Package):
    """A simple, scalable 3D Sn transport proxy app."""
    depends_on('mpi', when="+mpi")

    def install(self, spec, prefix):
        with working_dir('build', create=True):
            cmake('-DCMAKE_INSTALL_PREFIX=%s' % spec.prefix, '..',
                  *std_cmake_args)
            make()
            make('install')
```

Spack is used worldwide



12,343 web visits since May
Over **500** downloads per day

Over **200** contributors from labs, academia, industry.

Over **2,100** software packages.

Spack provides a *spec* syntax to describe customized DAG configurations

```
$ spack install mpileaks                              unconstrained
$ spack install mpileaks@3.3                          @ custom version
$ spack install mpileaks@3.3 %gcc@4.7.3               % custom compiler
$ spack install mpileaks@3.3 %gcc@4.7.3 +threads      +/- build option
$ spack install mpileaks@3.3 cppflags="-O3 -g3"       setting compiler flags
$ spack install mpileaks@3.3 os=CNL10 target=haswell  setting target for X-compile
$ spack install mpileaks@3.3 ^mpich@3.2 %gcc@4.9.3    ^ dependency information
```

- Each expression is a ***spec*** for a particular configuration
  - Each clause adds a constraint to the spec
  - Constraints are optional – specify only what you need.
  - Customize install on the command line!

- Spec syntax is recursive
  - Full control over the combinatorial build space

`spack list` shows what packages are available

```
$ spack list
==> 303 packages.
activeharmony   cgal              fish        gtkplus      libgd          mesa          openmpi          py-coverage   py-pycparser      qt          tcl
adept-utils     cgm               flex        harfbuzz     libgpg-error   metis         openspeedshop    py-cython     py-pyelftools     qthreads    texinfo
apex            cityhash          fltk        hdf          libjpeg-turbo  Mitos         openssl          py-dateutil   py-pygments       R           the_silver_searcher
arpack          cleverleaf        flux        hdf5         libjson-c      mpc           otf              py-epydoc     py-pylint         ravel       thrift
asciidoc        cloog             fontconfig  hwloc        libmng         mpe2          otf2             py-funcsigs   py-pypar          readline    tk
atk             cmake             freetype    hypre        libmonitor     mpfr          pango            py-genders    py-pyparsing      rose        tmux
atlas           cmocka            gasnet      icu          libNBC         mpibash       papi             py-gnuplot    py-pyqt           rsync       tmuxinator
atop            coreutils         gcc         icu4c        libpciaccess   mpich         paraver          py-h5py       py-pyside         ruby        trilinos
autoconf        cppcheck          gdb         ImageMagick  libpng         mpileaks      paraview         py-ipython    py-pytables       SAMRAI      uncrustify
automaded       cram              gdk-pixbuf  isl          libsodium      mrnet         parmetis         py-libxml2    py-python-daemon  samtools    util-linux
automake        cscope            geos        jdk          libtiff        mumps         parpack          py-lockfile   py-pytz           scalasca    valgrind
bear            cube              gflags      jemalloc     libtool        munge         patchelf         py-mako       py-rpy2           scorep      vim
bib2xhtml       curl              ghostscript jpeg         libunwind      muster        pcre             py-matplotlib py-scientificpython scotch      vtk
binutils        czmq              git         judy         libuuid        mvapich2      pcre2            py-mock       py-scikit-learn   scr         wget
bison           damselfly         glib        julia        libxcb         nasm          pdt              py-mpi4py     py-scipy          silo        wx
boost           dbus              glm         launchmon    libxml2        ncdu          petsc            py-mx         py-setuptools     snappy      wxpropgrid
bowtie2         docbook-xml       global      lcms         libxshmfence   ncurses       pidx             py-mysqldb1   py-shiboken       sparsehash  xcb-proto
boxlib          doxygen           glog        leveldb      libxslt        netcdf        pixman           py-nose       py-sip            spindle     xerces-c
bzip2           dri2proto         glpk        libarchive   llvm           netgauge      pkg-config       py-numexpr    py-six            spot        xz
cairo           dtcmp             gmp         libcerf      llvm-lld       netlib-blas   pmgr_collective  py-numpy      py-sphinx         sqlite      yasm
callpath        dyninst           gmsh        libcircle    lmdb           netlib-lapack postgresql       py-pandas     py-sympy          stat        zeromq
cblas           eigen             gnuplot     libdrm       lmod           netlib-scalapack ppl           py-pbr        py-tappy          sundials    zlib
cbtf            elfutils          gnutls      libdwarf     lua            nettle        protobuf         py-periodictable py-twisted     swig        zsh
cbtf-argonavis  elpa              gperf       libedit      lwgrp          ninja         py-astropy       py-pexpect    py-urwid          szip
cbtf-krell      expat             gperftools  libelf       lwm2           ompss         py-basemap       py-pil        py-virtualenv     tar
cbtf-lanl       extrae            graphlib    libevent     matio          ompt-openmp   py-biopython     py-pillow     py-yapf           task
cereal          exuberant-ctags   graphviz    libffi       mbedtls        opari2        py-blessings     py-pmw        python            taskd
cfitsio         fftw              gsl         libgcrypt    memaxes        openblas      py-cffi          py-pychecker  qhull             tau
```

- Spack has over 2,100 packages now.

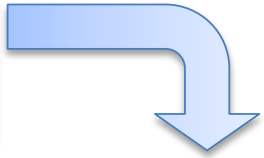
`spack find` shows what is installed

```
$ spack find
==> 103 installed packages.
-- linux-redhat6-x86_64 / gcc@4.4.7 ------------------------------
ImageMagick@6.8.9-10  glib@2.42.1        libtiff@4.0.3     pango@1.36.8          qt@4.8.6
SAMRAI@3.9.1          graphlib@2.0.0     libtool@2.4.2     parmetis@4.0.3        qt@5.4.0
adept-utils@1.0       gtkplus@2.24.25    libxcb@1.11       pixman@0.32.6         ravel@1.0.0
atk@2.14.0            harfbuzz@0.9.37    libxml2@2.9.2     py-dateutil@2.4.0     readline@6.3
boost@1.55.0          hdf5@1.8.13        llvm@3.0          py-ipython@2.3.1      scotch@6.0.3
cairo@1.14.0          icu@54.1           metis@5.1.0       py-nose@1.3.4         starpu@1.1.4
callpath@1.0.2        jpeg@9a            mpich@3.0.4       py-numpy@1.9.1        stat@2.1.0
dyninst@8.1.2         libdwarf@20130729  ncurses@5.9       py-pytz@2014.10       xz@5.2.0
dyninst@8.1.2         libelf@0.8.13      ocr@2015-02-16    py-setuptools@11.3.1  zlib@1.2.8
fontconfig@2.11.1     libffi@3.1         openssl@1.0.1h    py-six@1.9.0
freetype@2.5.3        libmng@2.0.2       otf@1.12.5salmon  python@2.7.8
gdk-pixbuf@2.31.2     libpng@1.6.16      otf2@1.4          qhull@1.0

-- linux-redhat6-x86_64 / gcc@4.8.2 ------------------------------
adept-utils@1.0.1  boost@1.55.0  cmake@5.6-special  libdwarf@20130729  mpich@3.0.4
adept-utils@1.0.1  cmake@5.6     dyninst@8.1.2      libelf@0.8.13      openmpi@1.8.2

-- linux-redhat6-x86_64 / intel@14.0.2 ---------------------------
hwloc@1.9  mpich@3.0.4  starpu@1.1.4

-- linux-redhat6-x86_64 / intel@15.0.0 ---------------------------
adept-utils@1.0.1  boost@1.55.0  libdwarf@20130729  libelf@0.8.13  mpich@3.0.4

-- linux-redhat6-x86_64 / intel@15.0.1 ---------------------------
adept-utils@1.0.1  callpath@1.0.2  libdwarf@20130729  mpich@3.0.4
boost@1.55.0       hwloc@1.9       libelf@0.8.13      starpu@1.1.4
```

- All the versions coexist!
  - Multiple versions of same package are ok.
- Packages are installed to automatically find correct dependencies.
- Binaries work *regardless of user's environment*.
- Spack also generates module files.
  - Don't *have* to use them.

Users can query the full dependency configuration of installed packages.

```
$ spack find callpath
==> 2 installed packages.
-- linux-x86_64 / clang@3.4 --------     -- linux-x86_64 / gcc@4.9.2 -------------
callpath@1.0.2                           callpath@1.0.2
```



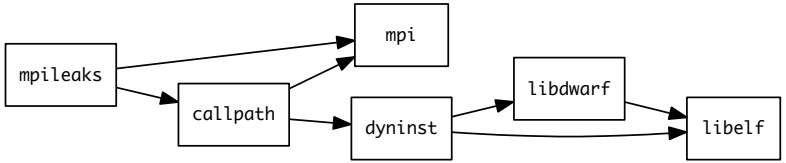
Expand dependencies with `spack find -d`

- Architecture, compiler, versions, and variants may differ between builds.

```
$ spack find -dl callpath
==> 2 installed packages.
-- linux-x86_64 / clang@3.4 -----------     -- linux-x86_64 / gcc@4.9.2 ----------
-
xv2clz2    callpath@1.0.2                    udltshs    callpath@1.0.2
ckjazss        ^adept-utils@1.0.1            rfsu7fb        ^adept-utils@1.0.1
3ws43m4            ^boost@1.59.0             ybet64y            ^boost@1.55.0
ft7znm6            ^mpich@3.1.4              aa4ar6i            ^mpich@3.1.4
qqnuet3        ^dyninst@8.2.1                tmnnge5        ^dyninst@8.2.1
3ws43m4            ^boost@1.59.0             ybet64y            ^boost@1.55.0
g65rdud            ^libdwarf@20130729        g2mxrl2            ^libdwarf@20130729
cj5p5fk                ^libelf@0.8.13        ynpai3j                ^libelf@0.8.13
cj5p5fk            ^libelf@0.8.13            ynpai3j            ^libelf@0.8.13
g65rdud        ^libdwarf@20130729            g2mxrl2        ^libdwarf@20130729
cj5p5fk            ^libelf@0.8.13            ynpai3j            ^libelf@0.8.13
cj5p5fk        ^libelf@0.8.13                ynpai3j        ^libelf@0.8.13
ft7znm6        ^mpich@3.1.4                  aa4ar6i        ^mpich@3.1.4
```

Spack handles combinatorial software complexity.

Dependency DAG



Installation Layout

Hash

```
spack/opt/
    linux-x86_64/
        gcc-4.7.2/
            mpileaks-1.1-0f54bf34cadk/
        intel-14.1/
            hdf5-1.8.15-lkf14aq3nqiz/
    bgq/
        xl-12.1/
            hdf5-1-8.16-fqb3a15abrwx/
    ...
```

- Each unique dependency graph is a unique ***configuration***.
- Each configuration installed in a unique directory.
  - Configurations of the same package can coexist.
- **Hash** of entire directed acyclic graph (DAG) is appended to each prefix.
- Installed packages automatically find dependencies
  - Spack embeds RPATHs in binaries.
  - No need to use modules or set LD_LIBRARY_PATH
  - Things work *the way you built them*

Spack manages installed compilers

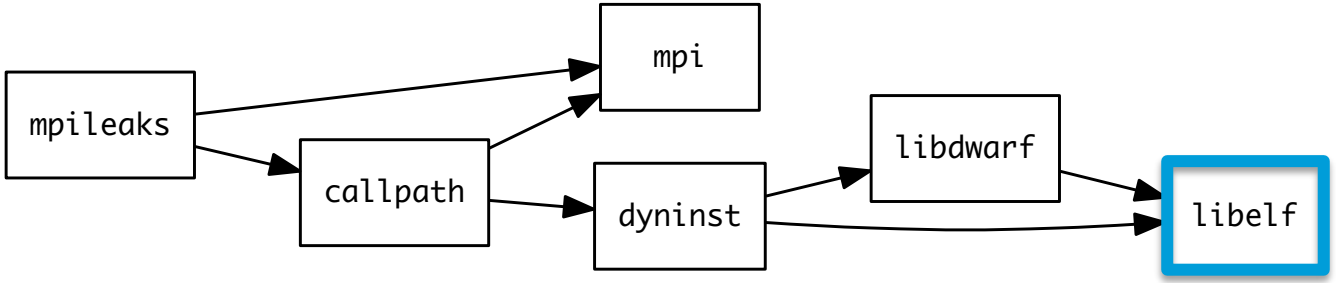
- Compilers are automatically detected
  - Automatic detection determined by OS
  - Linux: PATH
  - Cray: `module avail`
- Compilers can be manually added
  - Including Spack-built compilers

```
$ spack compilers
==> Available compilers
-- gcc ---------------------------------
gcc@4.2.1      gcc@4.9.3

-- clang -------------------------------
clang@6.0
```

compilers.yaml

```
compilers:
- compiler:
    modules: []
    operating_system: ubuntu14
    paths:
      cc: /usr/bin/gcc/4.9.3/gcc
      cxx: /usr/bin/gcc/4.9.3/g++
      f77: /usr/bin/gcc/4.9.3/gfortran
      fc: /usr/bin/gcc/4.9.3/gfortran
    spec: gcc@4.9.3
- compiler:
    modules: []
    operating_system: ubuntu14
    paths:
      cc: /usr/bin/clang/6.0/clang
      cxx: /usr/bin/clang/6.0/clang++
      f77: null
      fc: null
    spec: clang@6.0
- compiler:
    ...
```

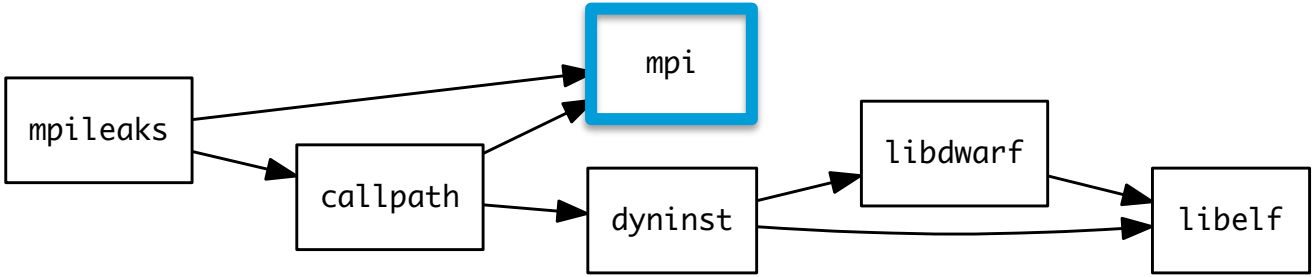
Spack Specs can constrain versions of dependencies




```
$ spack install mpileaks %intel@12.1 ^libelf@0.8.12
```

- Spack ensures *one* configuration of each library per DAG
  - Ensures ABI consistency.
  - User does not need to know DAG structure; only the dependency *names.*
- Spack can ensure that builds use the same compiler, or you can mix
  - Working on ensuring ABI compatibility when compilers are mixed.

Spack handles ABI-incompatible, versioned interfaces like MPI



- `mpi` is a *virtual dependency*
- Install the same package built with two different MPI implementations:

```
$ spack install mpileaks ^mvapich@1.9
```

```
$ spack install mpileaks ^openmpi@1.4:
```

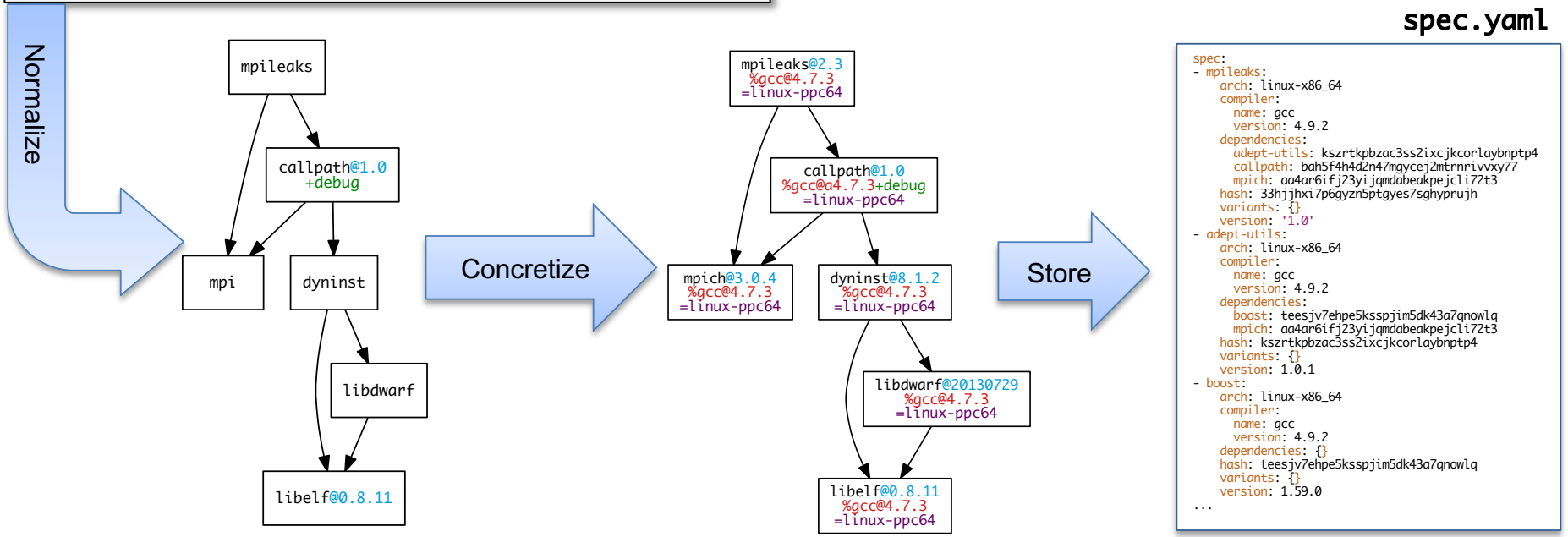
- Let Spack choose MPI implementation, as long as it provides MPI 2 interface:

```
$ spack install mpileaks ^mpi@2
```

Concretization fills in missing configuration details when the user is not explicit.

```
mpileaks ^callpath@1.0+debug ^libelf@0.8.11
```

User input: *abstract* spec with some constraints

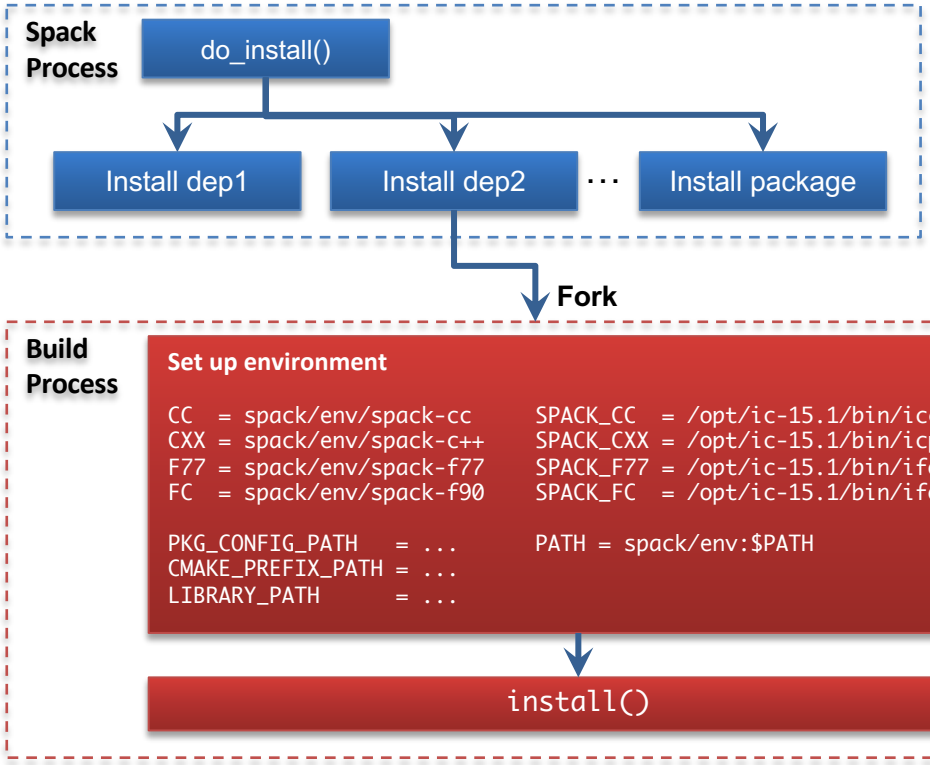


Abstract, normalized spec with some dependencies.

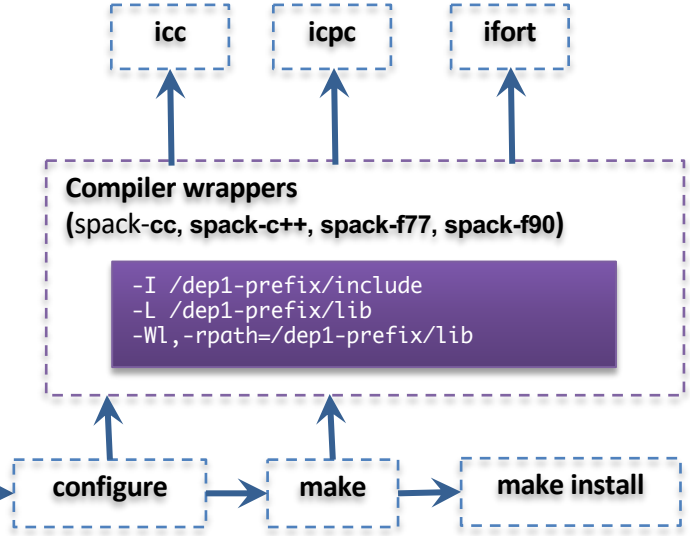
Concrete spec is fully constrained and can be passed to install.

Detailed provenance is stored with the installed package

Spack builds each package in its own compilation environment



- **Forked build process isolates environment for each build.**
  Uses compiler wrappers to:
  - Add include, lib, and RPATH flags
  - Ensure that dependencies are found automatically
  - Load Cray modules (use right compiler/system deps)



A note on reproducibility

- Spack builds can be reproduced from the same spec.yaml

```
spack install -f <spec.yaml>
```

- Builds are not completely isolated from the environment
  - Compiler wrappers help
  - We currently use blacklisting for environment variables
  - We would like to move to whitelisting, but platforms like Cray prevent this to some extent
    - Factory settings are inconsistent and brittle across sites.

- We're also looking into using Gentoo's jail mechanism
  - Doesn't require root; provides more reproducibility

Spack packages are templates

```
from spack import *

class Dyninst(Package):
    """API for dynamic binary instrumentation."""

    homepage = "https://paradyn.org"
    url = "http://www.paradyn.org/release8.1.2/DyninstAPI-8.1.2.tgz"

    version('8.2.1', 'abf60b7faabe7a2e')
    version('8.1.2', 'bf03b33375afa66f')
    version('8.1.1', 'd1a04e995b7aa709')

    depends_on("libelf")
    depends_on("libdwarf")
    depends_on("boost@1.42:")

    def install(self, spec, prefix):
        libelf = spec['libelf'].prefix
        libdwarf = spec['libdwarf'].prefix

        with working_dir('spack-build', create=True):
            cmake('..',
                  '-DBoost_INCLUDE_DIR=%s' % spec['boost'].prefix.include,
                  '-DBoost_LIBRARY_DIR=%s' % spec['boost'].prefix.lib,
                  '-DBoost_NO_SYSTEM_PATHS=TRUE'
                  *std_cmake_args)
            make()
            make("install")

    @when('@:8.1')
    def install(self, spec, prefix):
        configure("--prefix=" + prefix)
        make()
        make("install")
```

Metadata

Versions

Dependencies

Patches & variants (not shown)

Main install method

Optionally overload on spec constraints

Writing Packages - Versions and URLs

$REPO/packages/mvapich/package.py

```
class Mvapich2(Package):
    homepage = "http://mvapich.cse.ohio-state.edu/"
    url = "http://mvapich.cse.ohio-state.edu/download/mvapich/mv2/mvapich2-2.2rc2.tar.gz"

    version('2.2rc2', 'f9082ffc3b853ad1b908cf7f169aa878')
    version('2.2b',   '5651e8b7a72d7c77ca68da48f3a5d108')
    version('2.2a',   'b8ceb4fc5f5a97add9b3ff1b9cbe39d2')
    version('2.1',    '0095ceecb19bbb7fb262131cb9c2cdd6')
```

- Package downloads are hashed with MD5 by default
  - Also supports SHA-1, SHA-256, SHA-512
  - We'll be switching to SHA-256 or higher soon.
- Download URLs can be automatically extrapolated from URL.
  - Extra options can be provided if Spack can't extrapolate URLs
- Options can also be provided to fetch from VCS repositories

Writing Packages – Variants and Dependencies

$REPO/packages/petsc/package.py

```
class Petsc(Package):
    variant('mpi',     default=True,  description='Activates MPI support')
    variant('complex', default=False, description='Build with complex numbers')
    variant('hdf5',    default=True,  description='Activates support for HDF5 (only parallel)')

    depends_on('blas')
    depends_on('python@2.6:2.7')

    depends_on('mpi', when='+mpi')
```

- Variants are named, have default values and help text
- Other packages can be dependencies
  - **when** clause provides conditional dependencies
  - Can depend on specific versions or other variants

Writing Packages – Build Recipes

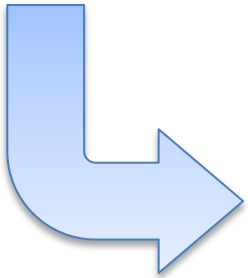
- Functions wrap common ops
  - cmake, configure, patch, make, …
  - **Executable** and **which** for new wrappers.
- Commands executed in clean environment
- Full Python functionality
  - Patch up source code
  - Make files and directories
  - Calculate flags
  - …

$REPO/packages/dyninst/package.py

```
def install(self, spec, prefix):
    with working_dir("build", create=True):
        cmake("..", *std_cmake_args)
        make()
        make("install")


@when('@:8.1')
def install(self, spec, prefix):
    configure("--prefix=" + prefix)
    make()
    make("install")
```

Create new packages with spack create

```
$ spack create http://zlib.net/zlib-1.2.8.tar.gz
```



$REPO/packages/zlib/package.py

```
class Zlib(Package):
    # FIXME: Add a proper url for your package's homepage here.
    homepage = "http://www.example.com"
    url      = "http://zlib.net/zlib-1.2.8.tar.gz"
    version('1.2.8', '44d667c142d7cda120332623eab69f40')

    def install(self, spec, prefix):
        # FIXME: Modify the cmake line to suit your build system here.
```

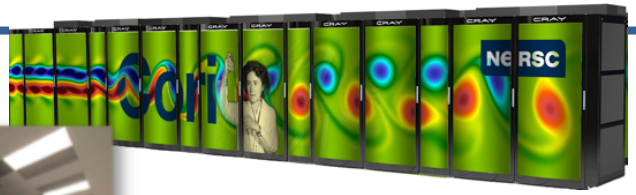
- spack create <url> will create a skeleton for a package
  - Spack reasons about URL, hash, version, build recipe.
  - Generates boilerplate for Cmake, Makefile, autotools, Python, R, Waf, Perl
    - Not intended to completely write the package, but gets you 80% of the way there.
- spack edit <package> for subsequent changes

github/spack/spack/releases/

We recently released Spack v0.11

- 2,178 packages (up from 1,114 a year ago)
- Big features for users:
  - Relocatable binary packages (spack buildcache)
  - Full support for Python 3
  - Improved module support; custom module templates using jinja2
- Many improvements for packagers:
  - Multi-valued variants
  - Test dependency type
  - Packages can patch their dependencies (not just themselves)
- Many speed improvements (to Spack itself)

Spack will enable us to deliver the ECP software stack.



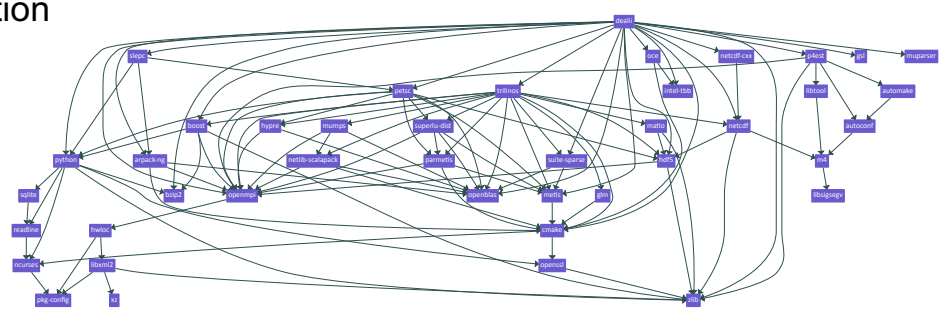
- Software in ECP stack needs to run on ECP platforms, testbeds, clusters, laptops
  - Each new environment requires effort.
- ECP asks us to build a software stack that will have an impact on industry.
  - Needs to be robust, tested, and reliable
  - Needs to be easy to get up and running
- We will provide the infrastructure necessary to make this tractable:
  1. A dependency model that can handle HPC software
  2. A hub for coordinated software releases (like xSDK)
  3. Build and test automation for large packages across facility
  4. Hosted binary and source software distributions for *all* ECP HPC platforms

Spack is the glue that holds the HPC software ecosystem together

github.com/spack

@spackpm

Package managers are increasingly integral to collaboration and reuse within developer communities



Hardly any HPC software package exists in isolation (though people may tell you otherwise)

Successes:

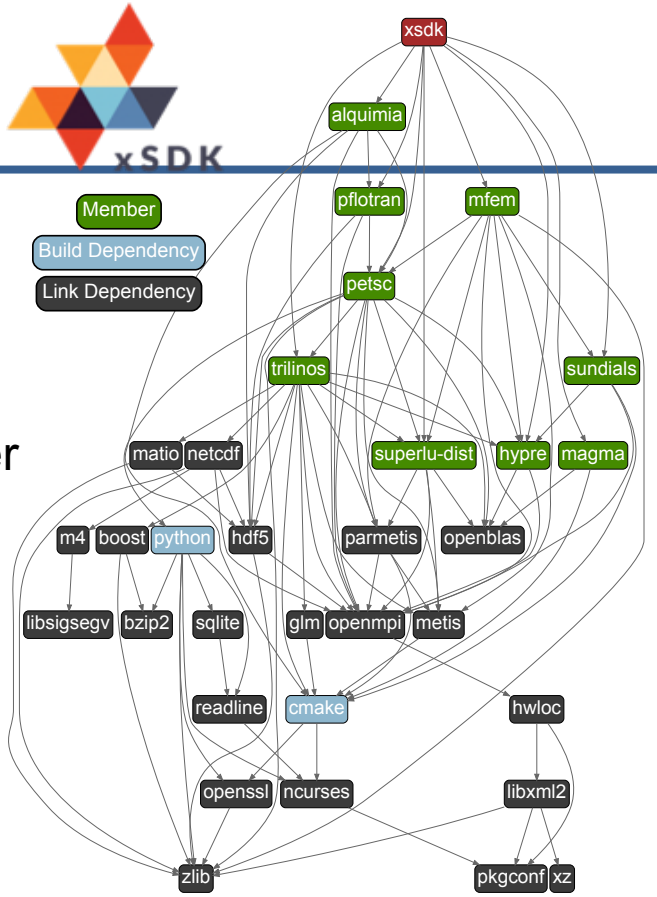
- xSDK has coordinated 2 releases with Spack
- ECP Proxy App Suite released using Spack
- QMCPACK, NALU, CEED stack, other releases
- ECP facilities are deploying software using Spack
- Spack is a requirement in the CORAL 2 RFP
- OLCF was able to build 192 packages for ARM ThunderX2 using Spack
- Spack used heavily outside ECP in the High Energy Physics (CERN, Fermilab), at EPFL, industry (ARM, Genentech), acadmia, and other institutions.
- SC tutorials have led to community members proposing a tutorial at ISC 2018

CEED EXASCALE DISCRETIZATIONS · Los Alamos NATIONAL LABORATORY · Fermilab · Lawrence Livermore National Laboratory · NERSC · PETSc · OAK RIDGE National Laboratory · ARM · CERN

xSDK · hypre · ECP Proxy App Suite · MFEM · Genentech · QMCPACK · Trilinos · Kitware · Argonne NATIONAL LABORATORY · openHPC · EPFL ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE · Sandia National Laboratories

The Spack team works with xSDK to manage releases

- Recently released xSDK 0.3.0 contains 8 member libraries, 22 required dependencies
- Coordination between Spack team and xSDK team allowed this to build at ANL, ORNL, NERSC, and other sites.
- First example of an ECP SDK distribution
  - Release process can serve as a guideline for teams implementing the new ST SDK directive.
  - Other communities have used Spack for multi-lib distributions before
    - INRIA MORSE solvers, CERN/Fermi HEP



The Spack team has developed new features for ECP projects when necessary

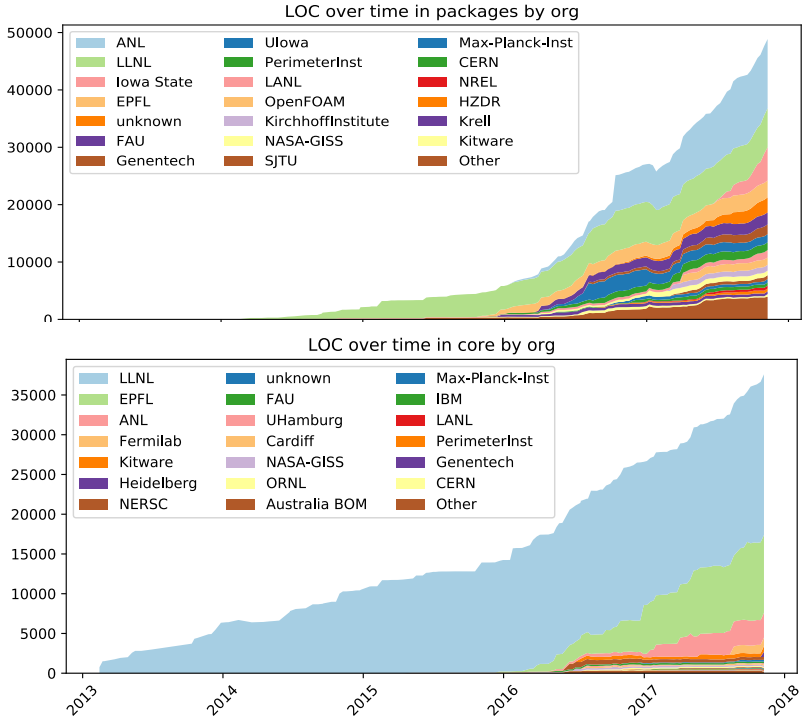
- Added ability for packages to maintain their own patches on dependencies, without disturbing other packages
  - QMCPACK maintains a 40,000-line patch on one of their dependencies
  - Flang also requires a patched LLVM installation
  - Allows patches not yet upstreamed to be maintained by patch authors

- Added a new *test dependency* type to allow QMCPACK team to only load some dependencies when running tests
  - We pulled items on our road map forward to make this happen

- DevRAMP team and Spack community provide feedback on pull requests from across the HPC ecosystem

Spack's community often delivers features *for us.*

- Discussions of ECP proxy applications on GitHub led to the idea of "tagging" packages
  - EPFL contributor added tagging capabilities to Spack *that day.*
- Examples of major external contributions:
  - Thousands of package recipes
  - Support for highly customized environment module generation
  - Multi-valued package build parameters
  - Build interface for BLAS/LAPACK/SCALAPACK
- Community members also help review code



Next Steps for Spack

1. **Better models for developer dependency management**
   - Top-level or free spackfile.yaml file in project
   - Developers to specify their dependency requirements by adding to the file
   - Spack automatically builds and sets up up environment for project
   - Environments can be shared with spackfile.lock file (Bundler-like model)

2. **We are building infrastructure for binary distribution**
   - Relocatable binary packaging capability complete, thanks to collaboration with Kitware, CERN, and Fermilab
   - Building and hosting of binary packages planned in Amazon S3
   - Requires new build bot infrastructure to manage binary builds and slice combinatorial space.

3. **We are developing a new concretizer for Spack**
   - SAT-based dependency resolution
   - Allows more aggressive reuse of installed packages
   - Allows dependency resolution with available binary packages
   - Allows more robust compiler dependency model


Lawrence Livermore
National Laboratory