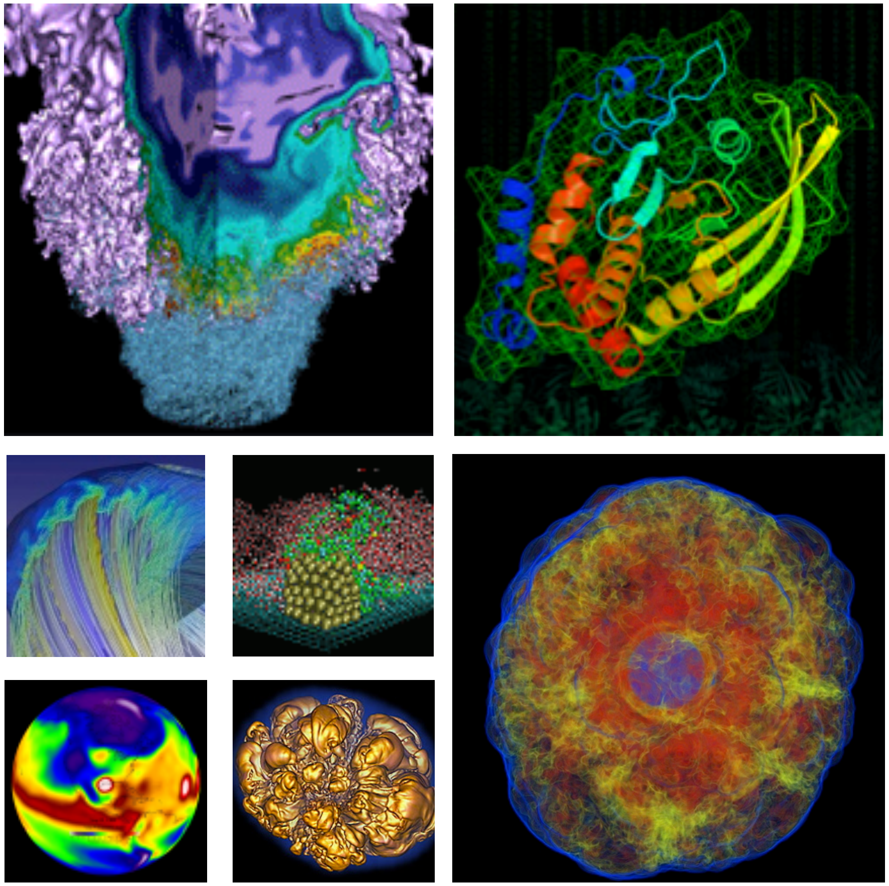
Using Spack to Manage Software on Cray Supercomputers



May 9th, 2017

Mario Melara (NERSC)
Todd Gamblin (LLNL)
Gregory Becker (LLNL)
Robert French (ORNL)
Matt P. Belhorn (ORNL)
Kelly Thompson (LANL)
Peter Scheibel (LLNL)
Rebecca Hartman-Baker (NERSC)

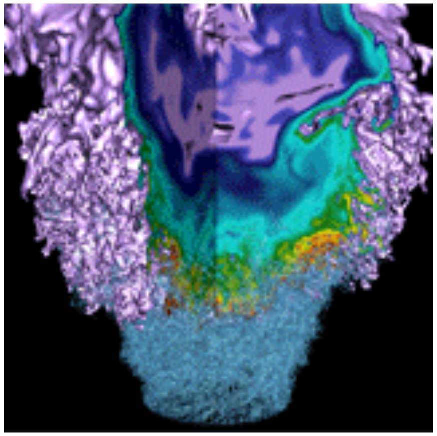
State of Software in HPC

- **Scientific software large and complex (multiple dependencies)**
- **HPC teams support different versions of software**
- **Same packages installed but built with different compilers/libraries/etc**

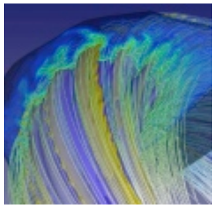
Tools for HPC Software

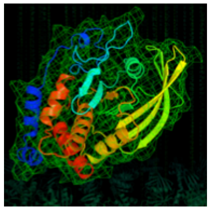
- **Package managers for HPC**
  - Smithy (ORNL)
  - SWTools (ORNL)
  - EasyBuild (UGhent)
  - Maali (Pawsey)
  - Spack (LLNL)

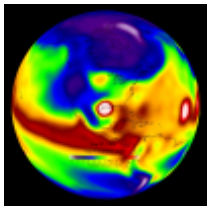
The Supercomputing PACKage Manager

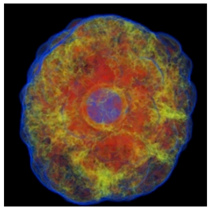


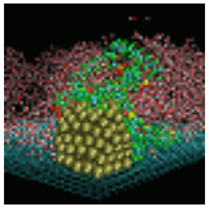
NERSC











SuperComputing PACKage Manager

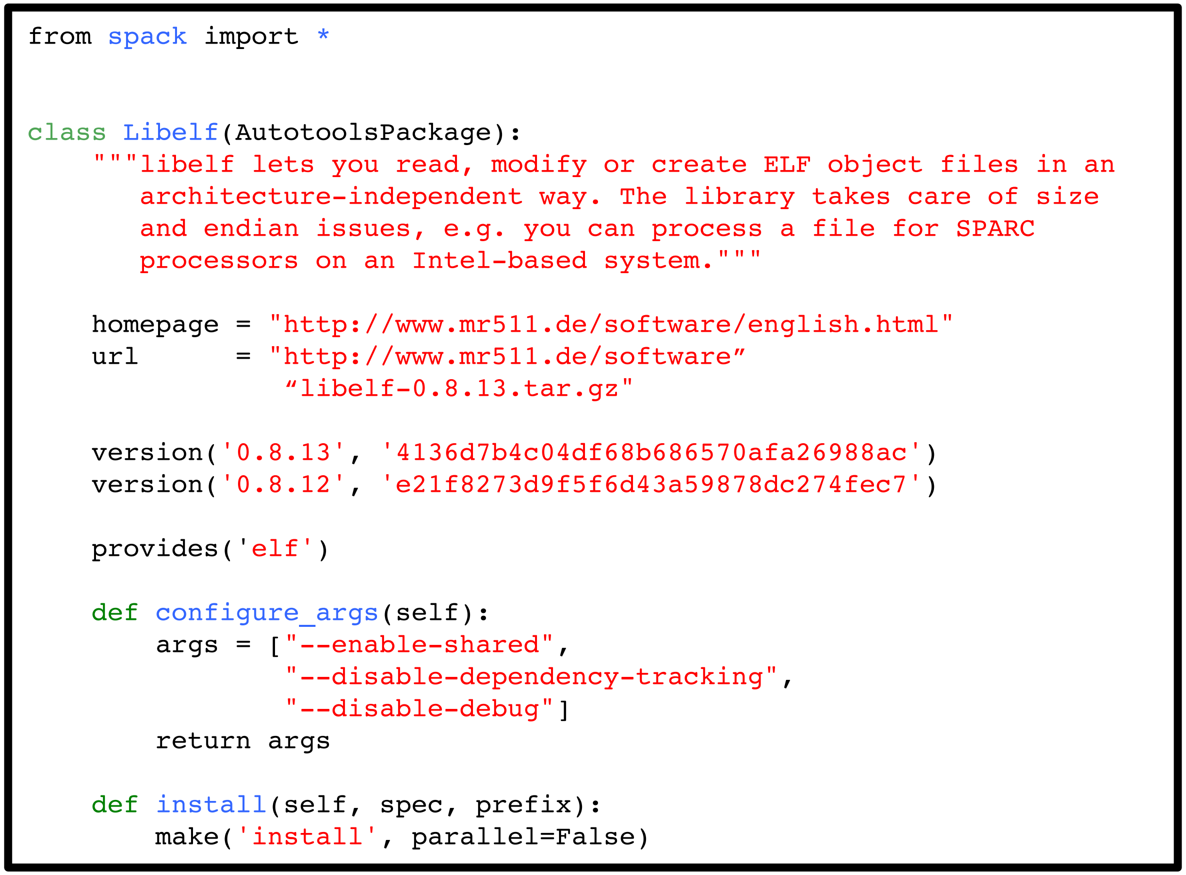
- **Spack included many features of interest to us at NERSC**
  - Easy to install and use
  - Packages are flexible
    - Build a range of versions for each package.
    - LLNL our close neighbors!
    - Todd Gamblin and Greg Becker (lead developers)
- **Lacked compatibility with our Cray systems**
  - Work needed to adapt Spack to Cray systems
  - Originally used on LLNL Linux clusters

Package template files

- **Allow different options to be chosen**
- **Can pick and choose dependencies, variants, compilers.**
- **Automatic patching**
- **Control over combinatorial space of a package.**
- **Wrapper for common file and build functions**
  - configure(), make()
- **Can modify build environment for package and it's dependents.**

Package template file

Example of a package template file



```
from spack import *


class Libelf(AutotoolsPackage):
    """libelf lets you read, modify or create ELF object files in an
       architecture-independent way. The library takes care of size
       and endian issues, e.g. you can process a file for SPARC
       processors on an Intel-based system."""

    homepage = "http://www.mr511.de/software/english.html"
    url      = "http://www.mr511.de/software”
                “libelf-0.8.13.tar.gz"

    version('0.8.13', '4136d7b4c04df68b686570afa26988ac')
    version('0.8.12', 'e21f8273d9f5f6d43a59878dc274fec7')

    provides('elf')

    def configure_args(self):
        args = ["--enable-shared",
                "--disable-dependency-tracking",
                "--disable-debug"]
        return args

    def install(self, spec, prefix):
        make('install', parallel=False)
```

Spack Spec Syntax

Spec Type	Spec Symbol	CLI Usage
package	package-name	python
version	@	python@2.7.13
compiler	%	python@2.7.13%gcc
architecture	arch=	python@2.7.13%gcc arch=cray-CNL-haswell
variant	+/-	python@2.7.13%gcc +tk arch=cray-CNL-haswell
compiler flags	ldflags=, cflags=, cxxflags=, cppflags=	python@2.7.13%gcc +tk arch=cray-CNL-haswell cflags=-02

Each parameter constrains the spec
Full control over the combinatorial space of a package.

$ spack spec blast-plus

- Spec represents a directed acyclic graph
- Concretization algorithm
  - Fixed-point

```
Blast-
plus@2.6.0%gcc@4.9.3+bzip2+freetype+gnutls+jpeg+lzo+openss
l+pcre+perl+png+python~static+zlib arch=cray-CNL-ivybridge
    ^bzip2@1.0.6%gcc@4.9.3+shared arch=cray-CNL-ivybridge
    ^freetype@2.7.1%gcc@4.9.3 arch=cray-CNL-ivybridge
        ^libpng@1.6.29%gcc@4.9.3 arch=cray-CNL-ivybridge
            ^zlib@1.2.11%gcc@4.9.3+pic+shared arch=cray-
CNL-ivybridge
        ^pkg-config@0.29.2%gcc@4.9.3+internal_glib
arch=cray-CNL-ivybridge
    ^gnutls@3.5.10%gcc@4.9.3 arch=cray-CNL-ivybridge
        ^nettle@3.2%gcc@4.9.3 arch=cray-CNL-ivybridge
            ^gmp@6.1.2%gcc@4.9.3 arch=cray-CNL-ivybridge
                ^m4@1.4.18%gcc@4.9.3+sigsegv arch=cray-
CNL-ivybridge
                    ^libsigsegv@2.11%gcc@4.9.3 arch=cray-
CNL-ivybridge
    ^jpeg@9b%gcc@4.9.3 arch=cray-CNL-ivybridge
    ^lzo@2.09%gcc@4.9.3 arch=cray-CNL-ivybridge
    ^openssl@1.0.2k%gcc@4.9.3 arch=cray-CNL-ivybridge
    ^pcre@8.40%gcc@4.9.3+utf arch=cray-CNL-ivybridge
    ^perl@5.24.1%gcc@4.9.3+cpanm arch=cray-CNL-ivybridge
        ^gdbm@1.13%gcc@4.9.3 arch=cray-CNL-ivybridge
    ^python@2.7.13%gcc@4.9.3~tk~ucs4 arch=cray-CNL-
ivybridge
        ^ncurses@6.0%gcc@4.9.3~symlinks arch=cray-CNL-
ivybridge
        ^readline@7.0%gcc@4.9.3 arch=cray-CNL-ivybridge
        ^sqlite@3.18.0%gcc@4.9.3 arch=cray-CNL-ivybridge
```

Installing Packages w/ Spack

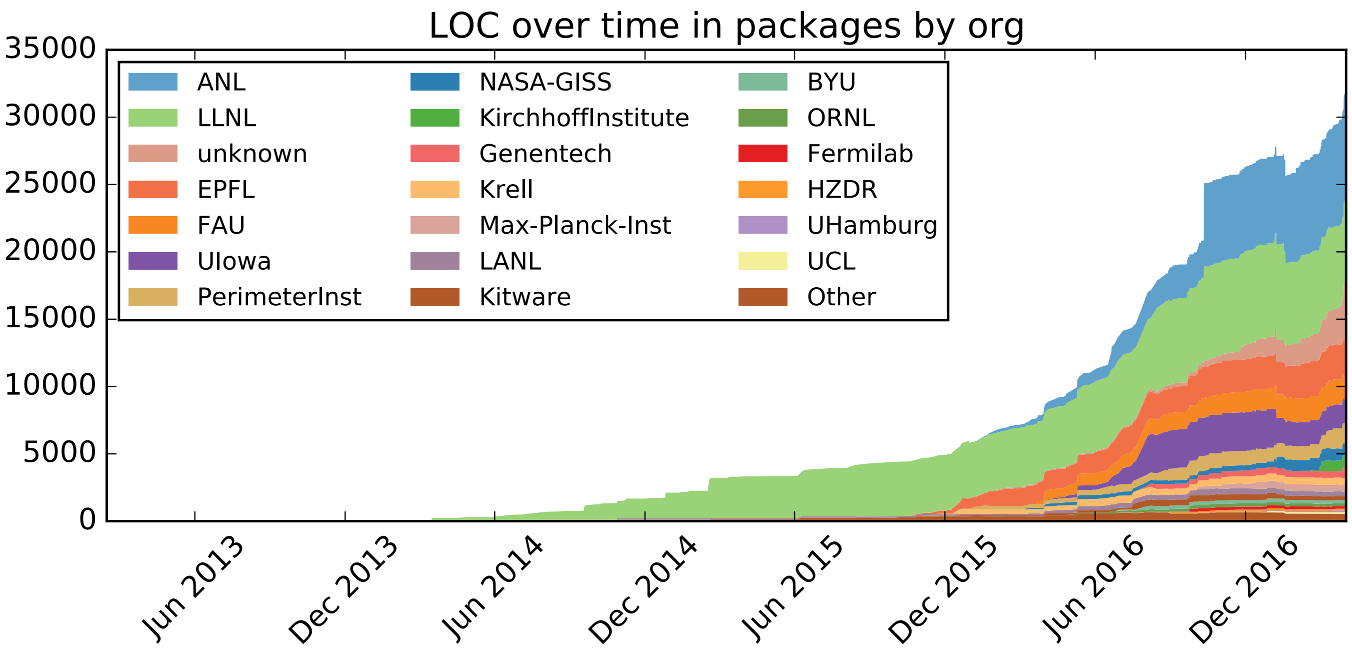
- **Easy as `spack install mpileaks`**
- **Spack compiler wrappers handle lib, include and RPATHs, and compiler flags.**
- **Each package has unique DAG-hash for provenance**
  - Directory tree generated for you
- **Generates modulefiles in a post-hook after install method completed.**
  - Supports Lmod, Dotkit, TCL

Spack Configuration

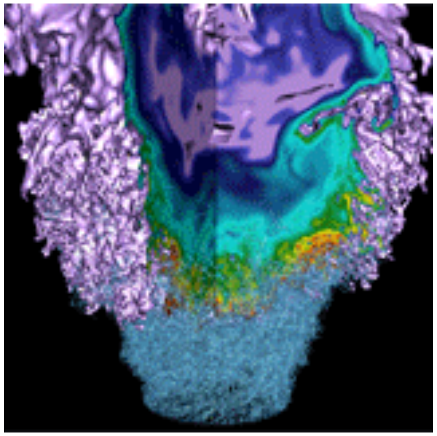
- **Spack allows flexibility and customization**
- **Configuration files**
  - packages.yaml
  - modules.yaml
  - compilers.yaml
  - config.yaml

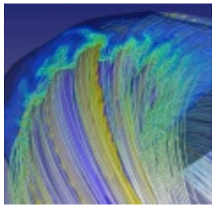
Spack Community

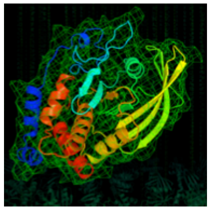
- When presented at SC ’15 Spack supported only 300 packages. Now over 1,400 packages are supported.
- Spack is in US Exascale Computing Project.
- Over 75% of packages are contributed externally.

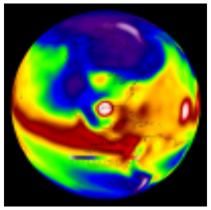


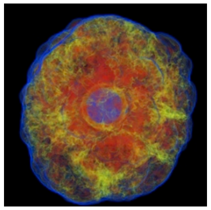
Cray Programming Environment

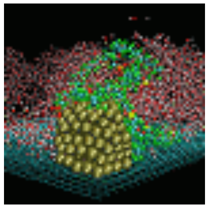












Cray Programming Environment

- **NERSC houses two Cray-XCs (Edison and Cori)**
- **Cori named top 5 most powerful supercomputer**
  - as of November 2016
- **Cori Architecture**
  - Intel Xeon (Haswell) and Intel Xeon Phi (Knight's Landing)
  - Burst Buffer
  - 250 Unique packages maintained on these systems

TCL Modules Environment

- **Dynamically modify the user's environment**
- **Vital for compiling codes**
  - Swap in and out of Programming Environments
  - Load necessary Cray optimized libraries.
  - Load target processors for cross-compiling.

NERSC default modules

```
Default Modules Found on Cori
1) modules/3.2.10.5             12) xpmem/2.1.1_gf9c9084-2.38
2) nsg/1.2.0                    13) job/2.1.1_gc1ad964-2.175
3) intel/17.0.2.174             14) dvs2.7_2.1.6
4) craype-network-aries         15) alps/6.3.4-2.21
5) craype/2.5.7                 16) rca/2.1.6_g2c60fbf-2.265
6) cray-libsci/16.09.1          17) atp/2.0.3
7) udreg/2.3.2-7.54             18) PrgEnv-intel/6.0.3
8) ugni/6.0.15-2.2              19) craype-haswell
9) pmi/5.0.10-1.0000           20) cray-shmem/7.4.4
10) dmapp/7.1.1-39.37           21) cray-mpich/7.4.4
11) gni-headers/5.0.11-2.2
```

Programming Environment Modules

- **Three types of PrgEnv modules**
  - PrgEnv-intel
  - PrgEnv-gnu
  - PrgEnv-cray
- **Load corresponding compiler: *Intel, GCC, CCE***
- **Compiler module controls compiler version.**

Target Modules

- **Cray machines are heterogeneous structures**
  - Front-end (login node) and back-end (compute nodes) vary in processor and operating system.
- **Front-end (login nodes)**
  - Basic tasks (I/O, filesystem tasks, loading data, compiling)
- **Back-end (compute nodes)**
  - High performance tasks
  - What your software should be optimized against
  - Special target flags in cray compilers placed based on loaded target.
- **Target modules exist so you can cross-compile without submitting a job.**

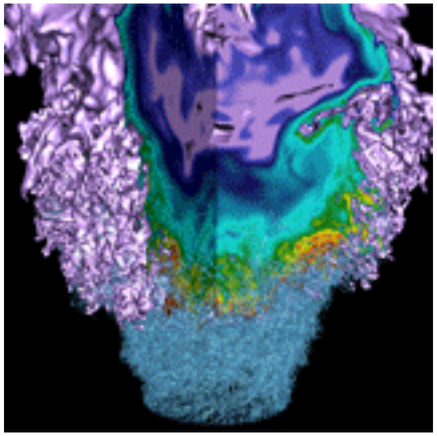
Cray and Third-Party Software

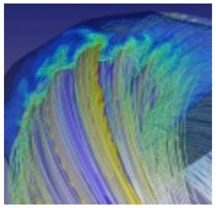
- **Cray provides optimized software libraries**
  - MPI (cray-mpich)
  - I/O Libraries (cray-netcdf, cray-hdf5)
  - Math libraries (cray-petsc, cray-trilinos)
  - Numerical routines (BLAS/LAPACK/SCALAPACK)
    - Cray-libsci, Intel-MKL
- **Monthly upgrades**
- **Libraries necessary for compiled software.**

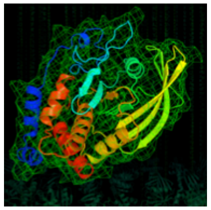
Cray Compiler Wrappers

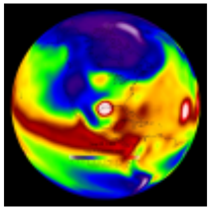
- **Long complex line of –L and –I flags.**
- **Contain optimization flags for architecture and target processor.**
- **Recommended to compile code for compute nodes.**
- **Requires compiling with Cray provided executables**
  - cc, CC, ftn

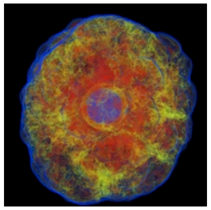
Spack on Cray

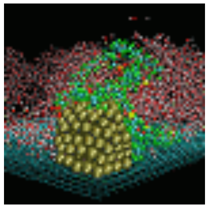












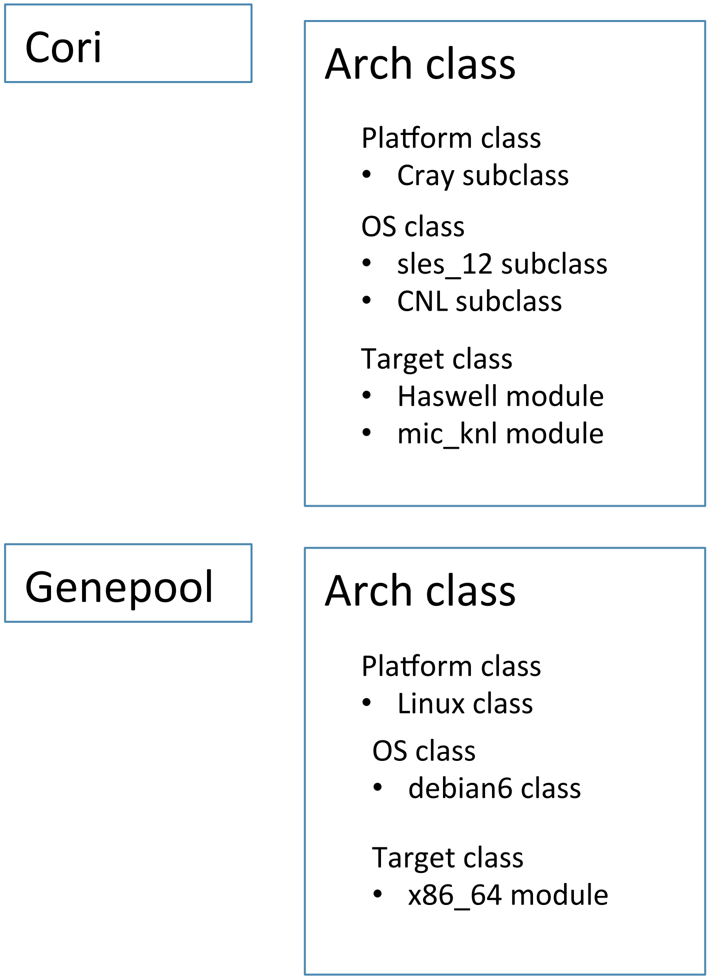
Spack on Cray

- **NERSC and LLNL collaboration to port Spack onto Cray.**
  - Module support
  - Architecture Spec
  - Compiler Detection/Wrapper Handling
  - Use of Cray packages
  - MPI support
  - Static and Dynamic Linking

Modules Support

- **Prior to v0.10 Spack could load own modules but not system modules.**
- **Python wrapper for modulecmd**
  - Takes as argument as shell name i.e Python, Ruby, Bash
  - Output can be parsed by that shell
- **Can load and unload modules**
  - PrgEnv-gnu/intel/cce
  - Target modules.

Architecture Spec



Cori

Arch class

Platform class
- Cray subclass

OS class
- sles_12 subclass
- CNL subclass

Target class
- Haswell module
- mic_knl module

Genepool

Arch class

Platform class
- Linux class

OS class
- debian6 class

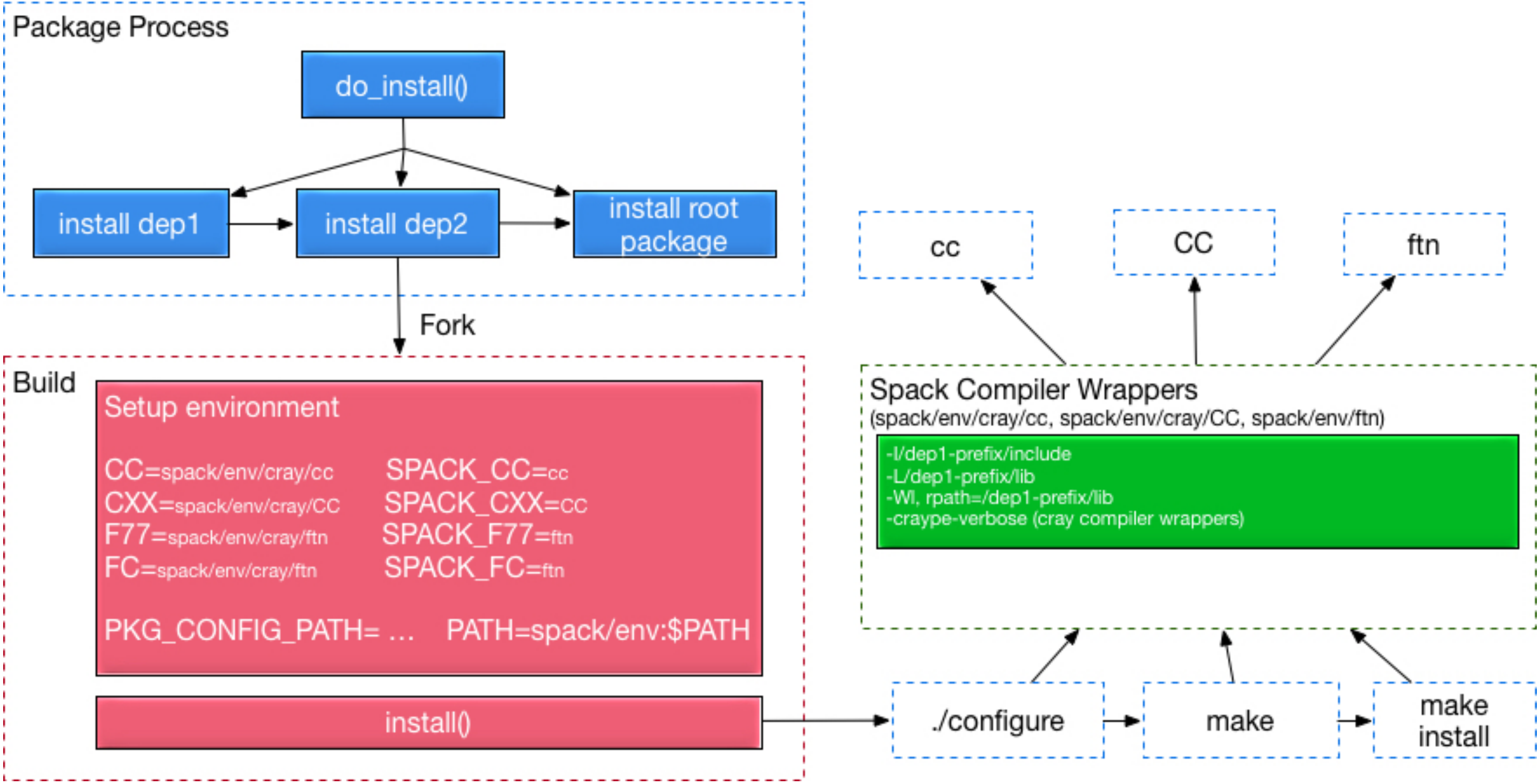
Target class
- x86_64 module

- **Previous versions of arch spec were just strings**
- **Spack uniquely provides support for compiling against different architectures.**
- **Auto-detected**
- **Cori defaults:**
  - arch=cray-CNL-haswell
- **Genepool defaults:**
  - arch=linux-debian6-x86_64

Improved Compiler Detection and Wrapper Handling

- Detection previously made through $PATH.
- modulecmd python avail (gcc/intel/cce)
- Point wrappers to cc, CC, ftn rather than direct compiler path.
- Add operating system/target to compiler meta-data.

Spack Compiler Wrappers



Support for Cray Packages

- **Existing logic already present in packages.yaml**
- **Declare packages as modules**
  - Spack can load the required module
  - Deduce path from modules

packages.yaml

```
packages:
    mpich:
        buildable: false
        modules:
            mpich@7.3.2%intel@17.0.0.098: cray-mpich/7.3.2
            mpich@7.4.1%cce@8.4.4: cray-mpich/7.4.1
            mpich@7.4.1%gcc@6.1.0: cray-mpich/7.4.1
    python:
        buildable: false
        paths:
            python@2.7.12%gcc@6.1.0: /global/common/software/python
            python@2.7.12%intel@17.0.0.098: /global/common/software/python
            python@2.7.12%cce@8.4.4: /global/common/software/python
```

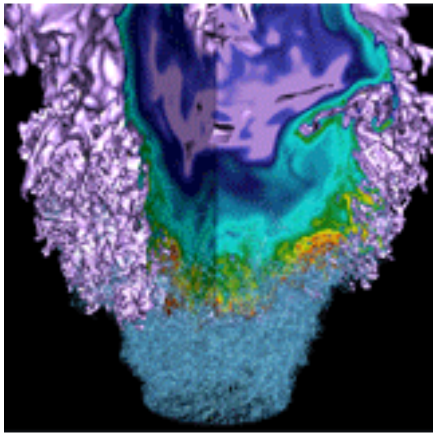
Message Passing Interface

- **Cray provides Module Passing Toolkit (MPT)**
  - cray-mpich, cray-shmem
- **MPI Compiler: cc, CC, ftn.**
- **User chooses how to compile**
  - Register MPI cray-mpich in packages.yaml.
  - Use Spack built MPI
- **Package writing for MPI is made simple**
  - env[‘CC’] = spec[‘mpi’].mpicc
  - env[‘CXX’] = spec[‘mpi’].mpicxx

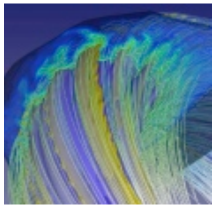
Static and Dynamic Linking

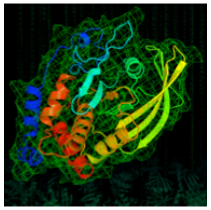
- **Default static linking**
  - Causes problems with most build system assumptions.
- **$CRAYPE_LINK_TYPE=dynamic**
  - Cray builds work like Linux builds
  - Spack doesn’t preclude static linking
    - Static flags added by build system.

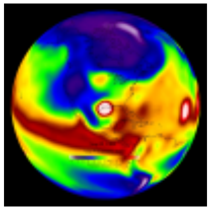
Spack at NERSC and ORNL

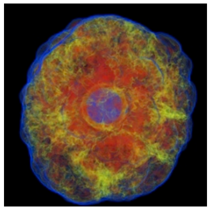


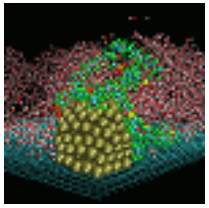
NERSC











NERSC Usage

- **Pseudo-user SWOWNER**
  - Login as pseudo-user and install packages
  - Special read and write
- **NERSC own github repo**
  - Forked from LLNL branch
  - Change our fork and merge changes into main repo
- **Can use SWOWNER Spack or your own**
- **Future plans to offer Spack as module to users**

ORNL Usage

- **Mainly used at OLCF and NCRC**
- **Number of packages installed with Spack**
  - 7/193 Titan (Cray XK7)
  - 5/73 Eos
- **On non-Cray 48/59 packages installed w/ Spack.**
- **Single Spack instance used per host.**
- **Future use to allow users to use Spack with limited permissions.**

Results at NERSC

- **Setting up Spack can be burdensome**
  - Hard to make configurations for all platforms
- **Platform specific compilers needed**
  - Most sites support multiple platforms
- **Modulefiles needs better customization**
  - No logic to switch between programming environments
- **Installation directory tree needs customization**
  - Most sites have canonical path for installs
- **Minimal stack on Edison/Cori**
  - Not ready for production. Yet!
- **ORNL wants to move to production as well!**

Results at ORNL

- Spack mostly used on non-Cray systems
- Move towards Spack on Cray is slow.
- Lower number of packages present on Titan and Eos.

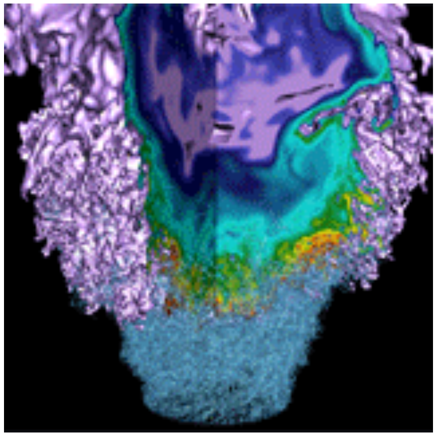
Related Package Managers

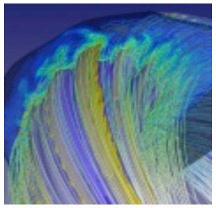
- **EasyBuild – Ghent University (Used at most CUG sites)**
  - Easyconfigs, easyblocks
- **Maali – Pawsey Supercomputing**
  - Collection of bash scripts
- **Smithy – ORNL**
  - Port based package manager specifically for Crays

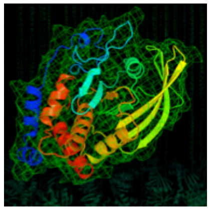
Comparison

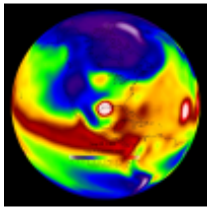
	EasyBuild	Spack	Maali	Smithy
Installs dependencies	Y	Y	N	N
Combinatorial Stack	Y	Y	N	N
Programming Language	Python	Python	Bash	Ruby
Generates modulefiles	Y	Y	Y	Y
Manipulates Cray modules	Y	Y	Y	Y
Dependency Resolution	N	Y	N	N
Version	v3.2.0 (stable)	V0.10 (alpha)	release_1.x	v1.6.5

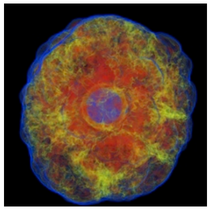
Conclusions

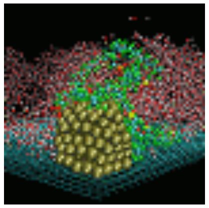












Conclusions

- **Spack is alpha software**
- **Future work still needed on Spack to accommodate our needs**
  - Default system package configuration
  - Modulefile logic for swapping PrgEnvs
  - Custom install naming scheme (PR submitted!)
  - Getting static builds is hard!
- **Package builds are unstable**
  - NERSC plans on nightly tests on Cray
- **Spack support for containers in Shifter.**
  - Use Spack as a package manager in images
- **Adopting a new tool is a slow process!**

NERSC

National Energy Research Scientific Computing Center