Spack: A Flexible Package Manager for HPC

BOF: Getting HPC Software Installed

SC'14, New Orleans, LA

http://bit.ly/spack-git

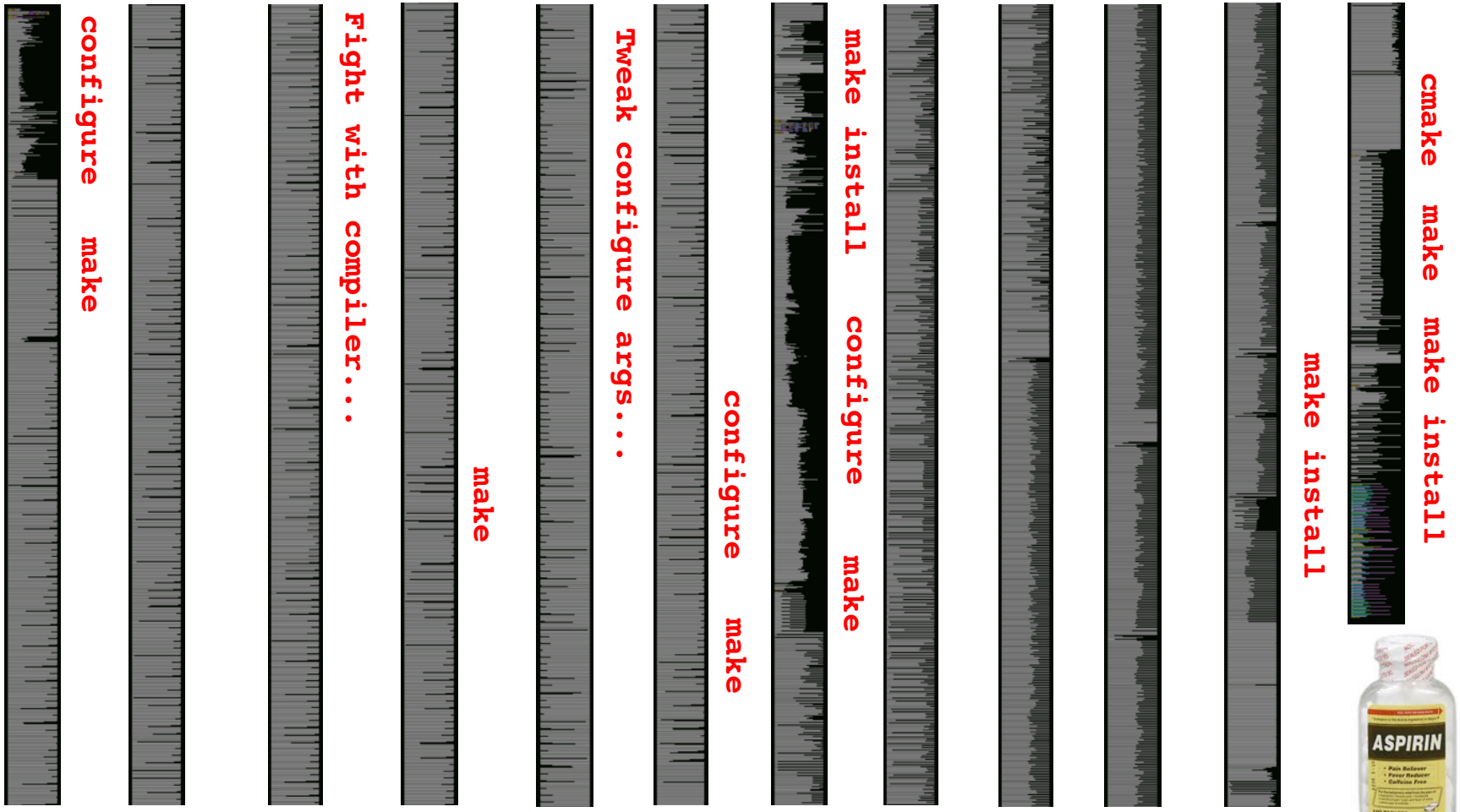
Todd Gamblin
Center for Applied Scientific Computing

Lawrence Livermore National Laboratory

LLNL-PRES-652881

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Building & installing software on HPC systems is extremely complex



Why is this so hard?

- Not much standardization in HPC
- Every machine and app has a different software stack.
- We want exotic architectures, compilers, MPI versions, and ***performance***
  - Might *need* to experiment with, e.g., 7 compilers and 2 versions of PETSc

48 third party packages

x **3 MPI versions**
mvapich mvapich2 OpenMPI

x **3-ish Platforms**
Linux BlueGene Cray

x **Up to 7 compilers**
Intel GCC XLC Clang
PGI Cray Pathscale

x **Oh, and 2-3 versions of each**

= **~7,500** combinations

- OK, so we don't build **all** of these
  - Many combinations don't make sense
- We want an easy way to quickly sample the space
  - Build a configuration on demand!

Spack makes building & installing simple

- Download spack (no need to install):

```
$ git clone https://github.com/scalability-llnl/spack.git
$ cd spack/bin
```

- Install software packages like this:

```
$ ./spack install mpileaks
```

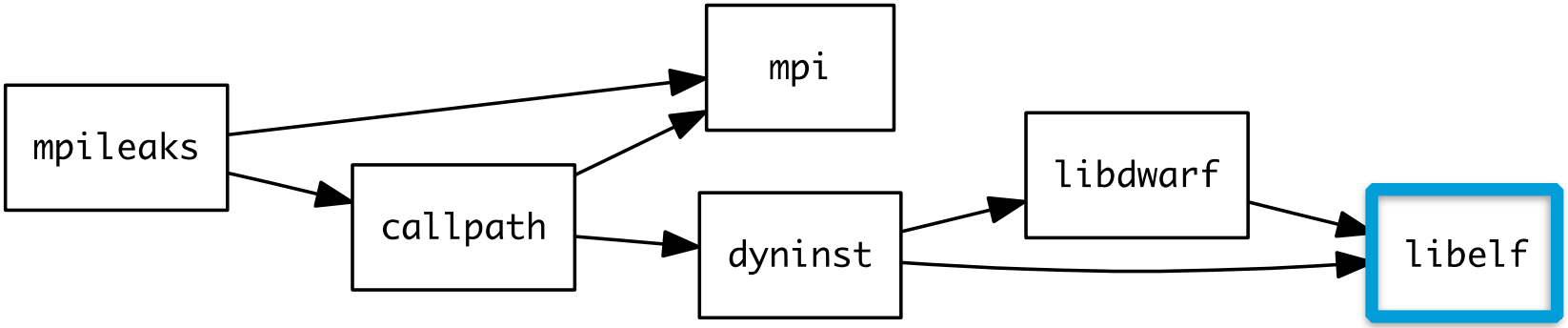
- This installs the **`mpileaks`** tool in spack/opt:
  1. Downloads the tarball
  2. Downloads any dependencies
  3. Builds and installs everything automatically
- **No root access required!**

Spack's *spec* syntax allows users to customize an installation

```
$ spack install mpileaks@1.1.2                  @ custom version
$ spack install mpileaks@1.1.2 %gcc@4.7.3       % custom compiler
$ spack install mpileaks@1.1.2 %gcc@4.7.3 +debug  +/- build option
$ spack install mpileaks@1.1.2 =bgqos_0         = cross-compile
```

- Each expression is a ***spec*** for a particular configuration
  - Clauses add constraints to the build
  - Constraints are optional – specify only what you need.
  - Spack fills in unspecified constraints with "sensible" defaults
    - Customizable policy for how to concretize abstract specs

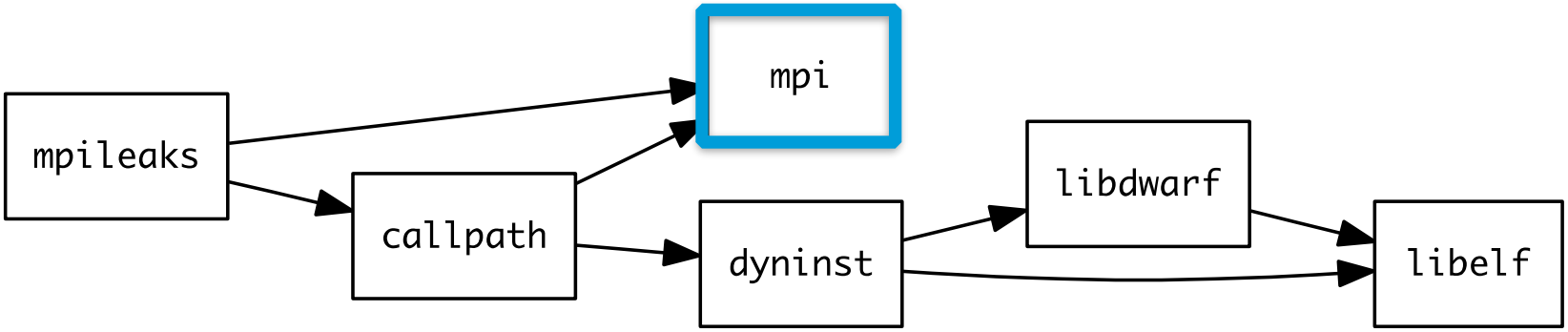
Specs can constrain dependency versions




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

- Spack ensures that all packages in the same install are built with the same version of libraries, like **`libelf`**.
- Spack can ensure that builds use the same compiler
  - Can also mix compilers but it's not default

Spack handles ABI incompatibility and versioned interfaces like MPI



Ask specifically for mvapich 1.9

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

Ask for openmpi 1.4 or higher

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

These install separately, in unique directories

Ask for an MPI that supports MPI-2 interface

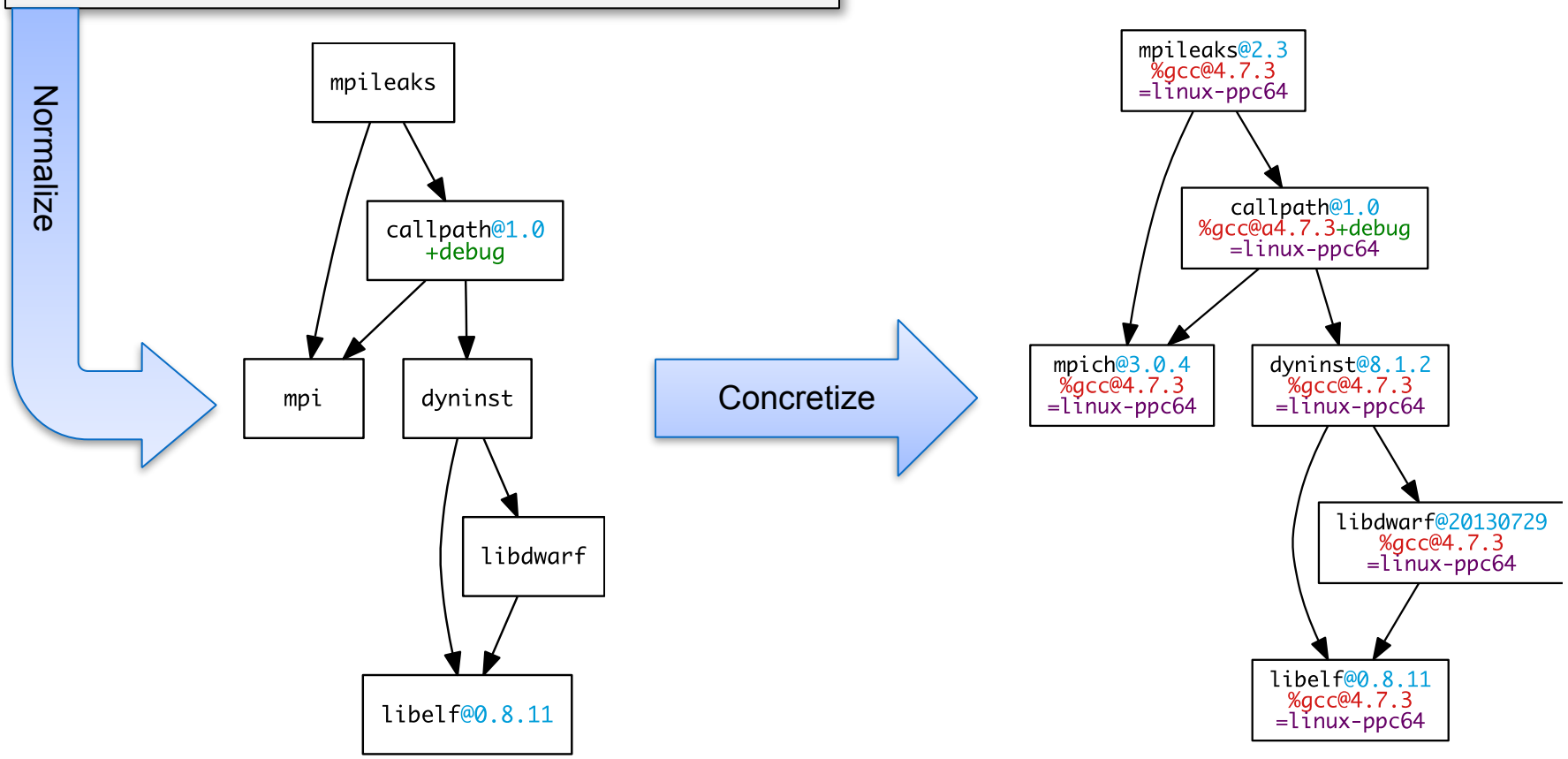
```
$ spack install mpileaks ^mpi@2
```

Spack chooses an MPI version that satisfies constraint

Spack fills in the blanks for the user

User input: *abstract* spec

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



Abstract, normalized spec has all dependencies.

Concrete spec is fully constrained and can be built.

Creating Spack packages is easy

```
$ spack create https://github.com/lee218llnl/stat/archive/v2.0.0.tar.gz
```

- Generates boilerplate package.py from a URL
- Automatically downloads and checksums archive

```
class Stat(Package):
    """FIXME: put a proper description of your package here."""
    homepage = "http://www.example.com"
    url      = "https://github.com/lee218llnl/stat/archive/v2.0.0.tar.gz"

    version('2.9.0b', '87bce8469240dc775c6c622c5f68fa87')

    def install(self, spec, prefix):
        configure("--prefix=%s" % prefix)
        make()
        make("install")
```

Use specs to query for installed versions of software

Find installed mpileaks versions built with mvapich and gcc

```
$ spack find mpileaks ^mvapich %gcc
== linux-ppc64 ===
---- gcc@4.4.7 ---
     mpileaks@1.0 ^mvapich@1.8.2
     mpileaks@1.1 ^mvapich@1.9.1
```

- Spec syntax doubles as a query language

Interest in Spack is growing!

- **Current participants:**
  - LLNL, LANL, Sandia, ANL
  - Some interest from Kitware for ParaView, Python installations
  - Interest from Open|SpeedShop developers
  - External contributors (AWE, other students)
- **79 packages** (and growing!)
  - 12 packages contributed in one week by a LANL user
  - STAT, mpileaks, Dyninst, other tools used at LLNL
  - GCC, LLVM, Clang
- **Core features continue to be added:**
  - Optional dependencies, variants in progress
  - BG/Q, Cray support planned to start early 2015
  - 1.0 will be released once these are done.

Get Spack 0.8.11!

http://bit.ly/spack-git

Spack Presentation

Thursday, 2:30pm
Booth #233, Stand #5
Emerging Tech. Showcase


Lawrence Livermore
National Laboratory