Hashdist – Yet Another Desperate Attempt at Fixing Scientific Software Distribution

Dag Sverre Seljebotn
Ondřej Čertík
Chris Kees

Simula, January 24, 2013

`http://github.com/hashdist`

The Problem

HPC software distribution

What makes HPC so special?

HPC software distribution

What makes HPC so special?

- ▶ Only sysadmins have root (so no Debian/Ubuntu, RedHat)
  - ▶ others: virtualization

HPC software distribution

What makes HPC so special?

- ▶ Only sysadmins have root (so no Debian/Ubuntu, RedHat)
  - ▶ others: virtualization
- ▶ Common to compile specifically for CPU for optimal speed
  - ▶ others: don't care so much

HPC software distribution

What makes HPC so special?

- ▶ Only sysadmins have root (so no Debian/Ubuntu, RedHat)
  - ▶ others: virtualization
- ▶ Common to compile specifically for CPU for optimal speed
  - ▶ others: don't care so much
- ▶ Users lack computer skills (relative to what they do)
  - ▶ others: either performance isn't important, or own a datacenter

HPC software distribution

What makes HPC so special?

- ▶ Only sysadmins have root (so no Debian/Ubuntu, RedHat)
  - ▶ others: virtualization
- ▶ Common to compile specifically for CPU for optimal speed
  - ▶ others: don't care so much
- ▶ Users lack computer skills (relative to what they do)
  - ▶ others: either performance isn't important, or own a datacenter
- ▶ Fortran compilers from multiple vendors with incompatible ABI
  - ▶ others: Fortran unheard of, only one compiler vendor anyway, C ABI standard

HPC software distribution

What makes HPC so special?

- ▶ Only sysadmins have root (so no Debian/Ubuntu, RedHat)
  - ▶ others: virtualization
- ▶ Common to compile specifically for CPU for optimal speed
  - ▶ others: don't care so much
- ▶ Users lack computer skills (relative to what they do)
  - ▶ others: either performance isn't important, or own a datacenter
- ▶ Fortran compilers from multiple vendors with incompatible ABI
  - ▶ others: Fortran unheard of, only one compiler vendor anyway, C ABI standard
- ▶ ```
  $ ls /usr/mpi/*
  /usr/mpi/gcc:
  mvapich-1.2.0 mvapich-1.2.0-qlc mvapich2-1.7
  mvapich2-1.7-qlc openmpi-1.4.3 openmpi-1.4.3-qlc
  /usr/mpi/intel:
  mvapich-1.2.0-qlc mvapich2-1.7-qlc openmpi-1.4.3-qlc
  /usr/mpi/pgi:
  mvapich-1.2.0-qlc mvapich2-1.7-qlc openmpi-1.4.3-qlc
  ```

# Scientific Python

- Python applications/scripts have large dependency trees
    - Really a result of *better* software engineering practices...

# Scientific Python

- ▶ Python applications/scripts have large dependency trees
  - ▶ Really a result of *better* software engineering practices...
- ▶ Python packaging created by web developers who mostly write Python and some C...

# Scientific Python

- Python applications/scripts have large dependency trees
  - Really a result of *better* software engineering practices...
- Python packaging created by web developers who mostly write Python and some C...
- ...while scientific Python really a shell around C, C++, Fortran codes

# Scientific Python

- Python applications/scripts have large dependency trees
  - Really a result of *better* software engineering practices...
- Python packaging created by web developers who mostly write Python and some C...
- ...while scientific Python really a shell around C, C++, Fortran codes

# Sophisticated systems

- Debian/Ubuntu/RedHat/Gentoo: Need root access

# Sophisticated systems

- Debian/Ubuntu/RedHat/Gentoo: Need root access
- MacPorts etc.: Mac only

# Sophisticated systems

- Debian/Ubuntu/RedHat/Gentoo: Need root access
- MacPorts etc.: Mac only
- Gentoo Prefix: Experimental

# Sophisticated systems

- Debian/Ubuntu/RedHat/Gentoo: Need root access
- MacPorts etc.: Mac only
- Gentoo Prefix: Experimental
- Grand Unified Builder, 0install: Lack scientific packages

# Sophisticated systems

- Debian/Ubuntu/RedHat/Gentoo: Need root access
- MacPorts etc.: Mac only
- Gentoo Prefix: Experimental
- Grand Unified Builder, 0install: Lack scientific packages
- Python environment provided by sysadmins: Outdated

# Simple user-space systems

Sage, python-hpcmp, dorsal, EasyBuild, Hans Petter's Python script:

- Simple (good) ⇔ lack features (minor inconvenience)

# Simple user-space systems

Sage, python-hpcmp, dorsal, EasyBuild, Hans Petter's Python script:

- Simple (good) ⇔ lack features (minor inconvenience)
- Easy to do oneself ⇒ difficult for one to get momentum

# Simple user-space systems

Sage, python-hpcmp, dorsal, EasyBuild, Hans Petter's Python script:

- Simple (good) ⇔ lack features (minor inconvenience)
- Easy to do oneself ⇒ difficult for one to get momentum
- The details are different for everybody
  - Best LAPACK for Sage is not best LAPACK for EPD/Anaconda

# Manual builds

- ./configure - -prefix=$HOME/local; make; make install

# Manual builds

- `./configure --prefix=$HOME/local; make; make install`
- Or in the case of my home institute, `export LD_LIBRARY_PATH=~oldstudent/local/lib`

# Manual builds

- ► `./configure --prefix=$HOME/local; make; make install`
- ► Or in the case of my home institute,
  `export LD_LIBRARY_PATH=~oldstudent/local/lib`

Some problems:

# Manual builds

- ► ./configure --prefix=$HOME/local; make; make install
- ► Or in the case of my home institute,
  export LD_LIBRARY_PATH=~oldstudent/local/lib

Some problems:

- ► Steep learning curve (and hard-to-find bugs)

# Manual builds

- ▶ ./configure --prefix=$HOME/local; make; make install
- ▶ Or in the case of my home institute,
  export LD_LIBRARY_PATH=~oldstudent/local/lib

Some problems:

- ▶ Steep learning curve (and hard-to-find bugs)
- ▶ Tedious to move between clusters

# Manual builds

- `./configure --prefix=$HOME/local; make; make install`
- Or in the case of my home institute,
  `export LD_LIBRARY_PATH=~oldstudent/local/lib`

Some problems:

- Steep learning curve (and hard-to-find bugs)
- Tedious to move between clusters
- Reproducibility

## The real knot

Combinatorial explosion:



MPI

Compiler vendor/version

LAPACK

$\times$ Python version $\times$ NumPy version $\times$ FFT library...

# Curation

- Debian, RedHat, cluster sysadmins, dorsal is all about curated software stacks

# Curation

- Debian, RedHat, cluster sysadmins, dorsal is all about curated software stacks
- Perhaps you want 60% curated, 20% bleeding edge or manually configured, 20% your own code...

# Curation

- Debian, RedHat, cluster sysadmins, dorsal is all about curated software stacks
- Perhaps you want 60% curated, 20% bleeding edge or manually configured, 20% your own code...
- Ubuntu + root access: Make your own PPA

# Curation

- Debian, RedHat, cluster sysadmins, dorsal is all about curated software stacks
- Perhaps you want 60% curated, 20% bleeding edge or manually configured, 20% your own code...
- Ubuntu + root access: Make your own PPA
- Build from source

# Curation

- Debian, RedHat, cluster sysadmins, dorsal is all about curated software stacks
- Perhaps you want 60% curated, 20% bleeding edge or manually configured, 20% your own code...
- Ubuntu + root access: Make your own PPA
- Build from source
- dorsal, EasyBuild: Contribute your own configuration
  - `dolfin.package` vs. `dolfin-intel.package`

# Hashdist

# Hash-based installation

- Linux laptop:
  `/usr/lib/libhdf5.so`

# Hash-based installation

- Linux laptop:
  ```
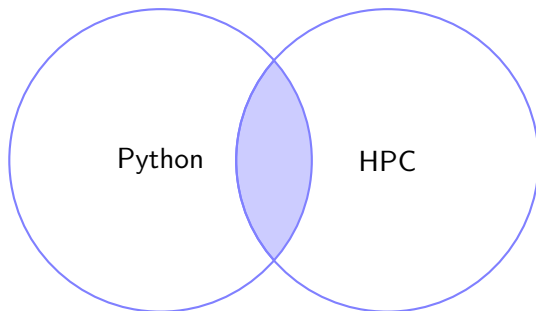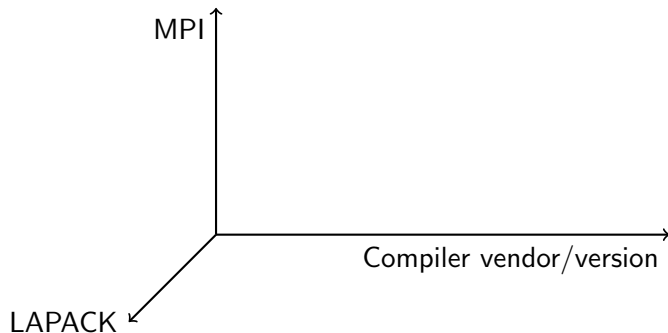 /usr/lib/libhdf5.so
  ```
- HPC environment modules:
  ```
  /cluster/software/VERSIONS/hdf5-1.6.1/lib/libhdf5.so
  /cluster/software/VERSIONS/hdf5-1.6.1_intel/lib/libhdf5.so
  /cluster/software/VERSIONS/hdf5-1.6.1_pgi/lib/libhdf5.so
  /cluster/software/VERSIONS/hdf5-1.8.9/lib/libhdf5.so
  /cluster/software/VERSIONS/hdf5-1.8.9_intel/lib/libhdf5.so
  /cluster/software/VERSIONS/hdf5-1.8.9_pgi/lib/libhdf5.so
  ```

# Hash-based installation

- Linux laptop:
  `/usr/lib/libhdf5.so`
- HPC environment modules:
  ```
  /cluster/software/VERSIONS/hdf5-1.6.1/lib/libhdf5.so
  /cluster/software/VERSIONS/hdf5-1.6.1_intel/lib/libhdf5.so
  /cluster/software/VERSIONS/hdf5-1.6.1_pgi/lib/libhdf5.so
  /cluster/software/VERSIONS/hdf5-1.8.9/lib/libhdf5.so
  /cluster/software/VERSIONS/hdf5-1.8.9_intel/lib/libhdf5.so
  /cluster/software/VERSIONS/hdf5-1.8.9_pgi/lib/libhdf5.so
  ```
- Hashdist:
  ```
  ~/.hdist/opt/hdf5/efn3/lib/libhdf5.so
  ~/.hdist/opt/hdf5/i7ni/lib/libhdf5.so
  ~/.hdist/opt/hdf5/qgpd/lib/libhdf5.so
  ```

  (really `hdf5/efn3i7ni7lbtik4frlb5wcnqgpdmi3ql`)

(demo)

## Using a hash-based software store

**(1)** Describe the build (**internal protocol!**)

```
{ "import" : [{ "id" : "gcc/apyicmxgafb564zz7rwhwvon7padvxdx"},
              { "id" : "unix/kmbvakqpvmhl65d2unzmjupxjbjkxge2"},
              { "id" : "zlib/wbg27phinbgwjg4nasb4xzf3ypo72otn"}],
   "sources" : [{ "key" : "tar.bz2:7jxgwn5xs5xnvsdaomvypridodr35or2"}],
   "script" : [
        ["LDFLAGS=-L${ZLIB}/lib -Wl,-rpath,${ZLIB}/lib"],
        ["CFLAGS=-I${ZLIB}/include"],
        ["./configure", "--prefix=${ARTIFACT}", "--with-pic"]
        ["make"],
        ["make", "install"]]
}
```

Using a hash-based software store

(1) Describe the build (**internal protocol!**)

```
{ "import" : [{ "id" : "gcc/apyicmxgafb564zz7rwhwvon7padvxdx"},
              { "id" : "unix/kmbvakqpvmhl65d2unzmjupxjbjkxge2"},
              { "id" : "zlib/wbg27phinbgwjg4nasb4xzf3ypo72otn"}],
  "sources" : [{ "key" : "tar.bz2:7jxgwn5xs5xnvsdaomvypridodr35or2"}],
  "script" : [
        ["LDFLAGS=-L${ZLIB}/lib -Wl,-rpath,${ZLIB}/lib"],
        ["CFLAGS=-I${ZLIB}/include"],
        ["./configure", "--prefix=${ARTIFACT}", "--with-pic"]
        ["make"],
        ["make", "install"]]
}
```

(2) Hash the build spec → `hdf5/u4vsabroylchvmwoxf5mdpxidd4lnrwl`

Using a hash-based software store

(1) Describe the build (**internal protocol!**)

```
{ "import" : [{ "id" : "gcc/apyicmxgafb564zz7rwhwvon7padvxdx"},
              { "id" : "unix/kmbvakqpvmhl65d2unzmjupxjbjkxge2"},
              { "id" : "zlib/wbg27phinbgwjg4nasb4xzf3ypo72otn"}],
  "sources" : [{ "key" : "tar.bz2:7jxgwn5xs5xnvsdaomvypridodr35or2"}],
  "script" : [
        ["LDFLAGS=-L${ZLIB}/lib -Wl,-rpath,${ZLIB}/lib"],
        ["CFLAGS=-I${ZLIB}/include -O0 -g"],
        ["./configure", "--prefix=${ARTIFACT}", "--with-pic"]
        ["make"],
        ["make", "install"]]
}
```

(2) Hash the build spec → `hdf5/fjczhadqtyx6jlbnvzlthrzsex7wz7xb`

Using a hash-based software store

(1) Describe the build (**internal protocol!**)

```
{ "import" : [{ "id" : "gcc/apyicmxgafb564zz7rwhwvon7padvxdx"},
              { "id" : "unix/kmbvakqpvmhl65d2unzmjupxjbjkxge2"},
              { "id" : "zlib/wbg27phinbgwjg4nasb4xzf3ypo72otn"}],
  "sources" : [{ "key" : "tar.bz2:7jxgwn5xs5xnvsdaomvypridodr35or2"}],
  "script" : [
      ["LDFLAGS=-L${ZLIB}/lib -Wl,-rpath,${ZLIB}/lib"],
      ["CFLAGS=-I${ZLIB}/include -O0 -g"],
      ["./configure", "--prefix=${ARTIFACT}", "--with-pic"]
      ["make"],
      ["make", "install"]]
}
```

(2) Hash the build spec → `hdf5/fjczhadqtyx6jlbnvzlthrzsex7wz7xb`

(3) If not found, do an *isolated* build on the fly

Using a hash-based software store

(4) When all packages are built reate symbolic links to package contents in a *profile*:

```
$ ls -l ~/local
local -> /home/dagss/.hdist/opt/profile/w6gp

$ ls -l /home/dagss/.hdist/opt/profile/w6gp/bin
h5copy -> /home/dagss/.hdist/opt/hdf5/whfk/bin/h5copy
h5ls -> /home/dagss/.hdist/opt/hdf5/whfk/bin/h5ls
...

$ ldd /home/dagss/.hdist/opt/hdf5/whfk/lib/libhdf5.so
linux-vdso.so.1 => (0x00007fffeb3ff000)
libsz.so.2 => /home/dagss/.hdist/opt/szip/5a5t/lib/libsz.so.2
libz.so.1 => /home/dagss/.hdist/opt/zlib/cll6/lib/libz.so.1
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
/lib64/ld-linux-x86-64.so.2
```

Consequences of hash-based installation

1. More dimensions! Even
   `.../h5py-hdf5_1.8.9_pgi-python2.7-numpy1.6.3_debug_ubuntu12.10`
   is not exhaustive; "`h5py/5ffg...`" caters for everything

Consequences of hash-based installation

1. More dimensions! Even
   `.../h5py-hdf5_1.8.9_pgi-python2.7-numpy1.6.3_debug_ubuntu12.10`
   is not exhaustive; "`h5py/5ffg...`" caters for everything
2. For free: Atomic upgrades

Consequences of hash-based installation

1. More dimensions! Even
   `.../h5py-hdf5_1.8.9_pgi-python2.7-numpy1.6.3_debug_ubuntu12.10`
   is not exhaustive; "`h5py/5ffg...`" caters for everything
2. For free: Atomic upgrades
3. Much easier: Uninstall (GC rather than file tracking)

Consequences of hash-based installation

1. More dimensions! Even
   `.../h5py-hdf5_1.8.9_pgi-python2.7-numpy1.6.3_debug_ubuntu12.10`
   is not exhaustive; "`h5py/5ffg...`" caters for everything
2. For free: Atomic upgrades
3. Much easier: Uninstall (GC rather than file tracking)
4. Jump around in software history (or between branches) in seconds!

Consequences of hash-based installation

1. More dimensions! Even
   `.../h5py-hdf5_1.8.9_pgi-python2.7-numpy1.6.3_debug_ubuntu12.10`
   is not exhaustive; "h5py/5ffg..." caters for everything
2. For free: Atomic upgrades
3. Much easier: Uninstall (GC rather than file tracking)
4. Jump around in software history (or between branches) in
   seconds!

Sophisticated features with simple implementation

Prior art: Eelco Dolstra's PhD thesis/the Nix project

The tradeoff: Builds must be "functional"

- All dependencies and all of the environment must be taken into account when describing the build

The tradeoff: Builds must be "functional"

- All dependencies and all of the environment must be taken into account when describing the build
- More hassle, but very good for reproducability

The tradeoff: Builds must be "functional"

- ▶ All dependencies and all of the environment must be taken into account when describing the build
- ▶ More hassle, but very good for reproducability

Some help:

The tradeoff: Builds must be "functional"

- All dependencies and all of the environment must be taken into account when describing the build
- More hassle, but very good for reproducability

Some help:

- Integrate with host system (Debian, environment modules, "generic") to specify dependencies on package on host system

The tradeoff: Builds must be "functional"

- All dependencies and all of the environment must be taken into account when describing the build
- More hassle, but very good for reproducability

Some help:

- Integrate with host system (Debian, environment modules, "generic") to specify dependencies on package on host system
- "hdist-jail" can issue warnings if a build process accesses files it shouldn't (or hide them)

Frontend: Software stack definitions (in progress)

Declarative approach (because you can git it and share it):

```
include:
  - sources # pull in ./sources.yml
  - build
  - when cluster == "abel":
    - abel-overrides
profiles:
  - name: "default"
    configuration:
      lapack_type: "openblas"
      cluster: "hexagon"
    select:
      - project: "hdf5"
        version: 1.8.2
      - project: "h5py"
        ...
```

Frontend: Software stack definitions (in progress)

Declarative approach (because you can git it and share it):

```
include:
  - sources # pull in ./sources.yml
  - build
  - when cluster == "abel":
    - abel-overrides
profiles:
  - name: "default"
    configuration:
      lapack_type: "openblas"
      cluster: "hexagon"
    select:
      - project: "hdf5"
        version: 1.8.2 to 1.8.5 # with integer linear programming
      - project: "h5py"
        ...
```

Domain-specific language focused on overrides

Manage the combinatorial explosion without creating packages for
hdf5_intel_mpich, hdf5_gcc_openmpi, ...:

```
rules:
  ...
  CFLAGS: ["-g", "-O$optlevel"]
  when recipe == "configure-make-install":
    optlevel: 2
  when project == "hdf5":
    recipe: "configure-make-install"
    when version == 1.5.2:
      optlevel: 0
    build_deps:
      - project: "zlib"
        version: 1.2.5 to 1.2.7
  ...
```

Build description pipeline

Results in a set of attributes per package:

```
dict(
 package='hdf5',
 version='1.8.10',
 recipe='configure-make-install',
 downloads=['http://www.hdfgroup.org/ftp/HDF5/current/'
            'src/hdf5-1.8.10.tar.bz2'],
 sources=['tar.bz2:7jxgwn5xs5xnvsdaomvypridodr35or2'],
 configure=['--prefix=$ARTIFACT', '--with-pic'],
 CFLAGS=['-O2'],
 jail='warn',
 build_deps=[zlib, unix, gcc]
)
```

Build description pipeline

Feed it through a Python pipeline:

```python
@pipeline.add_recipe('configure-make-install')
def configure_make_install_recipe(ctx, cfg, build):
    build['build']['script'].extend([
        ["LDFLAGS=$(hdist", "build-ldflags", ")"],
        ["CFLAGS=$(hdist", "build-cflags", ")"],
        ["CFLAGS+= " + " ".join(cfg.CFLAGS)],
        ['./configure'] + cfg.configure,
        ['make'],
        ['make', 'install']
        ])
```

Build description pipeline

To get build specification:

```
{ "import" : [{ "id" : "gcc/apyicmxgafb564zz7rwhwvon7padvxdx"},
              { "id" : "unix/kmbvakqpvmhl65d2unzmjupxjbjkxge2"},
              { "id" : "zlib/wbg27phinbgwjg4nasb4xzf3ypo72otn"}],
  "sources" : [{ "key" : "tar.bz2:7jxgwn5xs5xnvsdaomvypridodr35or2"]},
  "script" : [
        ["LDFLAGS=$(hdist", "build-ldflags", ")"],
        ["CFLAGS=$(hdist", "build-cflags", ")"],
        ["CFLAGS+= -O2"],
        ["./configure", "--prefix=${ARTIFACT}"]
        ["make"],
        ["make", "install"]]
}
```