# 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`

Dag Sverre's work for Simula Innovation is funded by a grant from the International Research Office, US Army Engineer Research and Development Center (BAA contract W911NF-12-1-0604)

## It's 2013, and most of the installation world is better off than 10 years ago

- Debian/RedHat

## It's 2013, and most of the installation world is better off than 10 years ago

- Debian/RedHat
- App stores

## It's 2013, and most of the installation world is better off than 10 years ago

- Debian/RedHat
- App stores
- Web apps

## What makes scientific codes so special?

- No root access

## What makes scientific codes so special?

- No root access
- Sometimes you **need** the very latest version

## What makes scientific codes so special?

- No root access
- Sometimes you **need** the very latest version
- Fortran/C++ instead of C/Java/.NET

## What makes scientific codes so special?

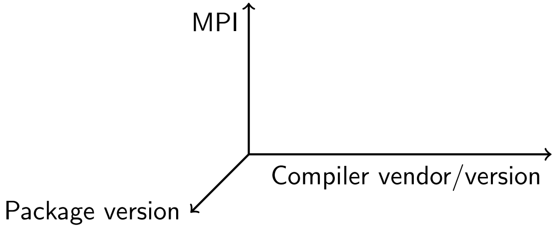
- No root access
- Sometimes you **need** the very latest version
- Fortran/C++ instead of C/Java/.NET
- Intersection of “need speed” and “do not pay dedicated application sysadmins”

## Combinatorial explosion

```
/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
```

## Combinatorial explosion

```
/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
```



$\times$ LAPACK $\times$ FFT library $\times$ IDL/Python version...

## Established options elsewhere

- Debian/Ubuntu/RedHat/Gentoo
  - Need root
  - Deals poorly with combinatorial explosion

## Established options elsewhere

- Debian/Ubuntu/RedHat/Gentoo
  - Need root
  - Deals poorly with combinatorial explosion
- MacPorts, Fink
  - Mac only
  - Deals “not good enough” with combinatorial explosion

## Established options elsewhere

- Debian/Ubuntu/RedHat/Gentoo
  - Need root
  - Deals poorly with combinatorial explosion
- MacPorts, Fink
  - Mac only
  - Deals "not good enough" with combinatorial explosion
- HPC environment modules
  - The sysadmins hate them
  - The users need newer/their own libraries

## Current solutions

- Build yourself from source

## Current solutions

- Build yourself from source
  - `./configure --prefix=$HOME/local; make; make install`

## Current solutions

- Build yourself from source
  - `./configure --prefix=$HOME/local; make; make install`
  - `export LD_LIBRARY_PATH=~sigurdkn/local/lib`

## Current solutions

- Build yourself from source
  - `./configure --prefix=$HOME/local; make; make install`
  - `export LD_LIBRARY_PATH=~sigurdkn/local/lib`
  - Fragile, duplication of work...

## Current solutions

- Build yourself from source
  - `./configure --prefix=$HOME/local; make; make install`
  - `export LD_LIBRARY_PATH=~sigurdkn/local/lib`
  - Fragile, duplication of work...
- Aha! Make a script to build from source...
  - Simula: dorsal
  - 5 different within DoD (incl. python-hpcmp)
  - Sage
  - Lots of others ("everybody" does it)

## Current solutions

- Build yourself from source
  - `./configure --prefix=$HOME/local; make; make install`
  - `export LD_LIBRARY_PATH=~sigurdkn/local/lib`
  - Fragile, duplication of work...
- Aha! Make a script to build from source...
  - Simula: dorsal
  - 5 different within DoD (incl. python-hpcmp)
  - Sage
  - Lots of others ("everybody" does it)
- Easy to do oneself $\Rightarrow$ difficult for one to get momentum

## Current solutions

- Build yourself from source
  - `./configure --prefix=$HOME/local; make; make install`
  - `export LD_LIBRARY_PATH=~sigurdkn/local/lib`
  - Fragile, duplication of work...
- Aha! Make a script to build from source...
  - Simula: dorsal
  - 5 different within DoD (incl. python-hpcmp)
  - Sage
  - Lots of others ("everybody" does it)
- Easy to do oneself $\Rightarrow$ difficult for one to get momentum
- The details are different for everybody

## 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 tweaked, 20% your own code...

- Others have failed, we're trying yet again

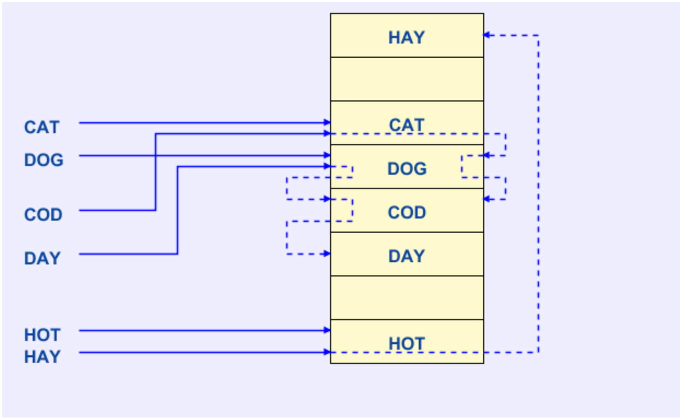
- Others have failed, we’re trying yet again
- Need to have new ideas

## Theory

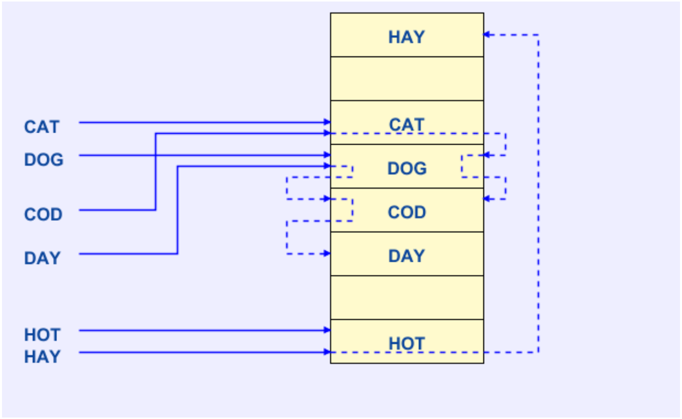
## Hash function

$$h(k) : \mathbb{N} \to \mathbf{H}$$

## Digression – $O(1)$ hash table table lookups

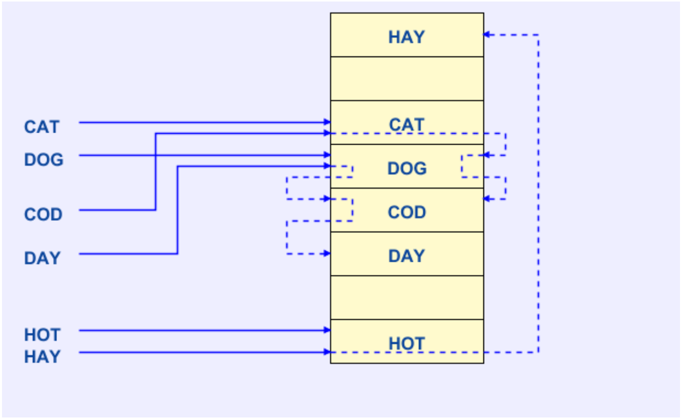


## Digression – $O(1)$ hash table table lookups



If you know contents up front: Create *perfect* $h$

## Digression – $O(1)$ hash table table lookups



If you know contents up front: Create *perfect* $h$

$h$ should be very fast; don't care so much about properties

Image: Hopgood (1968), Computer Bulletin

## Cryptographic/secure hashing

- Large key size (160, 256, or 512 bits)
    - NIST standards: SHA-1, SHA-2 (224, 256, 384, 512 bits), SHA-3

## Cryptographic/secure hashing

- Large key size (160, 256, or 512 bits)
  - NIST standards: SHA-1, SHA-2 (224, 256, 384, 512 bits), SHA-3
- Mathematical research topic to make $h$ such that no information about input is revealed

## Cryptographic/secure hashing

- Large key size (160, 256, or 512 bits)
  - NIST standards: SHA-1, SHA-2 (224, 256, 384, 512 bits), SHA-3
- Mathematical research topic to make $h$ such that no information about input is revealed

$h($`'The dark fox'`$) = h($546865206461726b20666f78 hex$)$
$=$ `b6589fc6ab0dc82cf12099d1c2d40ab994e8410c` hex

## Cryptographic/secure hashing

- Large key size (160, 256, or 512 bits)
  - NIST standards: SHA-1, SHA-2 (224, 256, 384, 512 bits), SHA-3
- Mathematical research topic to make $h$ such that no information about input is revealed

$h($`'The dark fog'`$) = h($`546865206461726b20666f67` hex$)$
$=$ `da4b9237bacccdf19c0760cab7aec4a8359010b0` hex

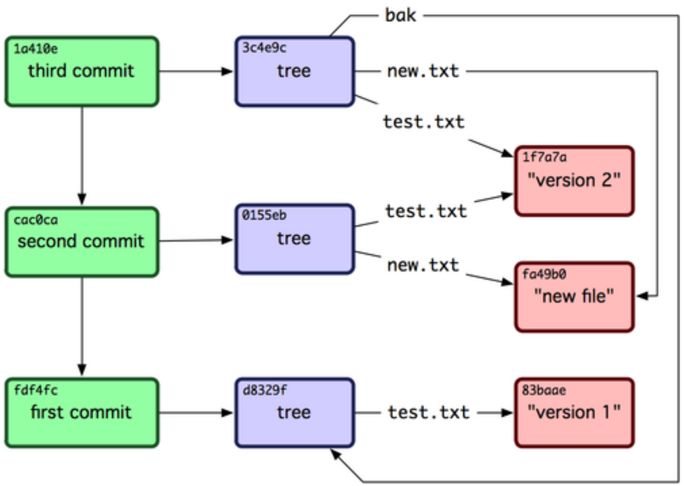
## Example 1: Store passwords

Use the one-way nature

## Example 2: git

```
$ (cd code/hashdist/.git/objects; find)
./59/5a2f8e3890d0ece24514f3e32ae874f1f03ac2
./2f/780151688e1f122a5b9072d42009c80c36140c
./2f/4b2eef40b51bc2d46027d1864653b37dd05f8f
./2f/237d74e3f81f498212629ac0b96bedac4b0b36
./2f/dff799c54fed6fe96a91e1d5f1593996228ebc
./2f/27bd4efa5f8521fb98eb82181a67aae97b7f1a
./2f/3fedf882f1b28905199961356f4e00281ddf76
```

## Example 2: git



## Why hashes?

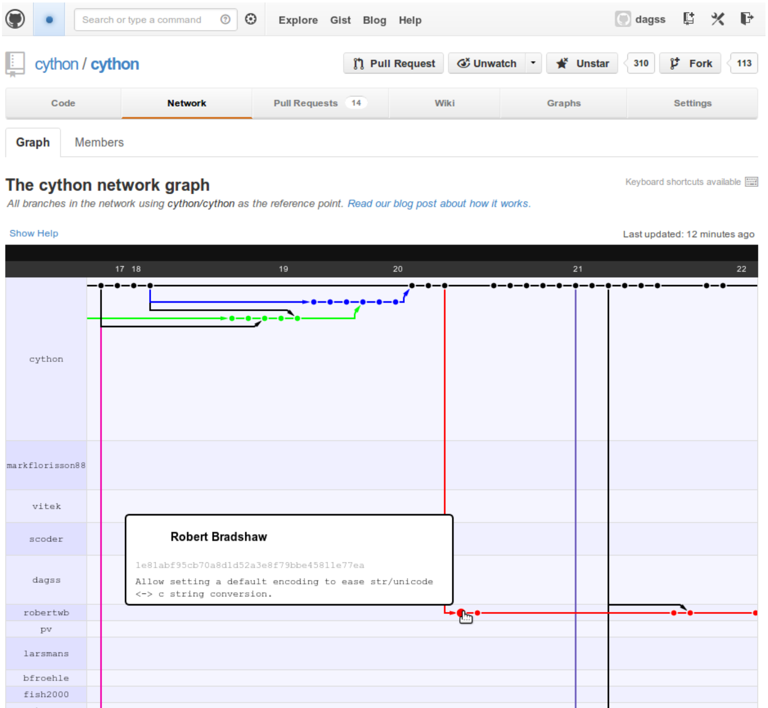
- Incremental database IDs don't work for distributed systems

## Why hashes?

- Incremental database IDs don't work for distributed systems
- Random ID's would work...
  - E.g., UUID like `550e8400-e29b-41d4-a716-446655440000`

## Why hashes?

- Incremental database IDs don't work for distributed systems
- Random ID's would work...
  - E.g., UUID like `550e8400-e29b-41d4-a716-446655440000`
- ...but a hash simultaneously verifies the content!



The cython network graph

*All branches in the network using* cython/cython *as the reference point. Read our blog post about how it works.*

Robert Bradshaw

1e81abf95cb70a8d1d52a3e8f79bbe45811e77ea

Allow setting a default encoding to ease str/unicode <-> c string conversion.

## Chances of collision...

Types of attack (assume $k = 160$):

## Chances of collision...

Types of attack (assume $k = 160$):

- Preimage attack: Chance $2^{-160} \sim 10^{-50}$

## Chances of collision...

Types of attack (assume $k = 160$):

- Preimage attack: Chance $2^{-160} \sim 10^{-50}$
- Accidental collision: May happen in a collection of $2^{80} \sim 10^{25}$ keys due to “birthday paradox”
    - 6.5 billion programmers...
    - ...each produce one Linux kernel history every second...
    - ...and push to one enourmous Git repsitory...
    - ...then after 5 years there’s a 50% chance of collision

## Chances of collision...

Types of attack (assume $k = 160$):

- Preimage attack: Chance $2^{-160} \sim 10^{-50}$
- Accidental collision: May happen in a collection of $2^{80} \sim 10^{25}$ keys due to "birthday paradox"
  - 6.5 billion programmers...
  - ...each produce one Linux kernel history every second...
  - ...and push to one enourmous Git repsitory...
  - ...then after 5 years there's a 50% chance of collision
- Brute-force SHA-1 still needs only $2^{60}$ operations due to weaknesses in SHA-1
  - $2.5M today, $50K after 2020.

## 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
  ```
  - Even `h5py-hdf5_1.8.9_pgi-python2.7-numpy1.6.3_abel_debug` incomplete

## 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
  ```
  - Even `h5py-hdf5_1.8.9_pgi-python2.7-numpy1.6.3_abel_debug` incomplete
  - Abel is new and small; US Army clusters have 20-30 versions of every package.

## 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
  ```
  - Even `h5py-hdf5_1.8.9_pgi-python2.7-numpy1.6.3_abel_debug` incomplete
  - Abel is new and small; US Army clusters have 20-30 versions of every package.
- Hashdist:
  ```
  ~/.hit/opt/hdf5/efn3/lib/libhdf5.so
  ~/.hit/opt/hdf5/i7ni/lib/libhdf5.so
  ~/.hit/opt/hdf5/qgpd/lib/libhdf5.so
  ```
  (really `hdf5/efn3i7ni7lbtik4frlb5wcnqgpdmi3ql`)

## Step 1: Hash the build

**Internal protocol!**

```
{
   "name": "hdf5",
   "import" : [{ "id" : "gcc/apyicmxgafb564zz7rwhwvon7padvxdx"},
               { "id" : "unix/v-1"},
               { "id" : "zlib/wbg27phinbgwjg4nasb4xzf3ypo72otn"}],
   "sources" : [{ "key" : "tar.bz2:7jxgwn5xs5xnvsdaomvypridodr35or2"}],
   "cmd": ["sh", "$in0"],
   "inputs": [
       "text": [
            ["./configure --prefix=${ARTIFACT} --with-pic \\"],
            ["    --with-zlib=$ZLIB"],
            ["make"],
            ["make", "install"]
        ]]
}
```

## Step 1: Hash the build

**Internal protocol!**

```
{
   "name": "hdf5",
   "import" : [{ "id" : "gcc/apyicmxgafb564zz7rwhwvon7padvxdx"},
               { "id" : "unix/v-1"},
               { "id" : "zlib/wbg27phinbgwjg4nasb4xzf3ypo72otn"}],
   "sources" : [{ "key" : "tar.bz2:7jxgwn5xs5xnvsdaomvypridodr35or2"}],
   "cmd": ["sh", "$in0"],
   "inputs": [
       "text": [
            ["./configure --prefix=${ARTIFACT} --with-pic \\"],
            ["     --with-zlib=$ZLIB"],
            ["make"],
            ["make", "install"]
       ]]
}
```

Hash to create key → `hdf5/u4vsabroylchvmwoxf5mdpxidd4lnrwl`

## Step 1: Hash the build

**Internal protocol!**

```
{
   "name": "hdf5",
   "import" : [{ "id" : "gcc/apyicmxgafb564zz7rwhwvon7padvxdx"},
               { "id" : "unix/v-1"},
               { "id" : "zlib/wbg27phinbgwjg4nasb4xzf3ypo72otn"}],
   "sources" : [{ "key" : "tar.bz2:7jxgwn5xs5xnvsdaomvypridodr35or2"}],
   "cmd": ["sh", "$in0"],
   "inputs": [
       "text": [
            ["./configure --prefix=${ARTIFACT} --with-pic \\"],
            ["    --with-zlib=$ZLIB CFLAGS=-O0"],
            ["make"],
            ["make", "install"]
        ]]
}
```

Hash to create key → `hdf5/fjczhadqtyx6jlbnvzlthrzsex7wz7xb`

## Step 2: Every build installs to separate location

Same as with the “module load“ system:

```
$ echo opt/*/*
opt/hdf5/avnj opt/hdf5/a5df opt/hdf5/a4sf opt/python/arxd
opt/python/rvdo opt/readline/6vvu  opt/readline/v7fw
opt/zlib/fh7n  opt/zlib/i7yr ...

$ ls opt/zlib/fh7n/lib
libz.a  libz.so  libz.so.1  libz.so.1.2.5  pkgconfig

$ ls opt/hdf5/avnj/lib
libhdf5.a  libhdf5.so  libhdf5.so.7  libhdf5.so.7.0.4
```

## Step 3: Make a profile with links

```
$ ls -la opt/profile/ldhn/bin
h5dump -> ../../../hdf5/avnj/bin/h5dump
h5import -> ../../../hdf5/avnj/bin/h5import
h5ls -> ../../../hdf5/avnj/bin/h5ls
...
```

## Branchable software stack

-
  ```
  ~/mystack $ ls
  default.yml sources.yml build.yml abel-cluster.yml
  ```

## Branchable software stack

-
  ```
  ~/mystack $ ls
  default.yml sources.yml build.yml abel-cluster.yml
  ```

-
  ```
  ~/mystack $ hit update # takes some time
  ~/mystack $ ~/local/bin/python --version
  Python 2.7.2
  ```

## Branchable software stack

-
  ```
  ~/mystack $ ls
  default.yml sources.yml build.yml abel-cluster.yml
  ```

-
  ```
  ~/mystack $ hit update # takes some time
  ~/mystack $ ~/local/bin/python --version
  Python 2.7.2
  ```

-
  ```
  ~/mystack $ hit update # instant
  ```

## Branchable software stack

-
  ```
  ~/mystack $ ls
  default.yml sources.yml build.yml abel-cluster.yml
  ```

-
  ```
  ~/mystack $ hit update # takes some time
  ~/mystack $ ~/local/bin/python --version
  Python 2.7.2
  ```

-
  ```
  ~/mystack $ hit update # instant
  ```

-
  ```
  ~/mystack $ git fetch from-someone-else
  ~/mystack $ git checkout someone-elses-branch
  ~/mystack $ hit update # pause for extra builds
  ~/mystack $ ~/local/bin/python --version
  Python 2.7.3
  ```

## Branchable software stack

-
  ```
  ~/mystack $ ls
  default.yml sources.yml build.yml abel-cluster.yml
  ```

-
  ```
  ~/mystack $ hit update # takes some time
  ~/mystack $ ~/local/bin/python --version
  Python 2.7.2
  ```

-
  ```
  ~/mystack $ hit update # instant
  ```

-
  ```
  ~/mystack $ git fetch from-someone-else
  ~/mystack $ git checkout someone-elses-branch
  ~/mystack $ hit update # pause for extra builds
  ~/mystack $ ~/local/bin/python --version
  Python 2.7.3
  ```

-
  ```
  ~/mystack $ git checkout paper-from-2010
  ~/mystack $ hit update # instant
  ~/mystack $ ~/local/bin/python --version
  Python 2.6.3
  ```

## 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

```
$ hit upgrade
# ...then power shuts down, but you’re good
```

## 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
   ```
   $ hit upgrade
   # ...then power shuts down, but you're good
   ```
3. Much easier: Uninstall (mark&sweep 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
   ```
   $ hit upgrade
   # ...then power shuts down, but you're good
   ```
3. Much easier: Uninstall (mark&sweep 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

   ```
   $ hit upgrade
   # ...then power shuts down, but you're good
   ```
3. Much easier: Uninstall (mark&sweep 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

## Ideas for autumn

- Specifying the entire software stack in minute detail is bothersome

## Ideas for autumn

- Specifying the entire software stack in minute detail is bothersome
- Want a constraint solver
  - A[version≥2] needs B[version≥1.2]
  - B[version≥1.1] needs LAPACK=intel-mkl
  - Given constraints, which LAPACK and which version of A, B

## Ideas for autumn

- Specifying the entire software stack in minute detail is bothersome
- Want a constraint solver
  - A[version$\geq$2] needs B[version$\geq$1.2]
  - B[version$\geq$1.1] needs LAPACK=intel-mkl
  - Given constraints, which LAPACK and which version of A, B
- Debian does constraint solving between packages
  - But we have $\prod_{i \in \{\text{MPI}, \text{compiler}, \ldots\}} n_i$ possible “packages”

## Ideas for autumn

- Specifying the entire software stack in minute detail is bothersome
- Want a constraint solver
  - A[version$\geq$2] needs B[version$\geq$1.2]
  - B[version$\geq$1.1] needs LAPACK=intel-mkl
  - Given constraints, which LAPACK and which version of A, B
- Debian does constraint solving between packages
  - But we have $\prod_{i \in \{\text{MPI},\text{compiler},\ldots\}} n_i$ possible "packages"
- We want to do constraint solving on input variables to the build

## Ideas for autumn

- Specifying the entire software stack in minute detail is bothersome
- Want a constraint solver
  - A[version$\geq$2] needs B[version$\geq$1.2]
  - B[version$\geq$1.1] needs LAPACK=intel-mkl
  - Given constraints, which LAPACK and which version of A, B
- Debian does constraint solving between packages
  - But we have $\prod_{i\in\{\text{MPI},\text{compiler},\ldots\}} n_i$ possible “packages”
- We want to do constraint solving on input variables to the build
- Objective function: Miminimize build time or maximize package freshness?

## Ideas for autumn

- Specifying the entire software stack in minute detail is bothersome
- Want a constraint solver
  - A[version≥2] needs B[version≥1.2]
  - B[version≥1.1] needs LAPACK=intel-mkl
  - Given constraints, which LAPACK and which version of A, B
- Debian does constraint solving between packages
  - But we have $\prod_{i \in \{\text{MPI},\text{compiler},\ldots\}} n_i$ possible “packages”
- We want to do constraint solving on input variables to the build
- Objective function: Miminimize build time or maximize package freshness?
- NP-complete problem, e.g.:
  - Integer linear programming
  - Belief propagation (from Bayesian network theory)

## Why bother?

- Leaving build input variables free (e.g., LAPACK) allows "re-seating" a software stack description from one system to another

## Why bother?

- Leaving build input variables free (e.g., LAPACK) allows "re-seating" a software stack description from one system to another
- Small step towards the "reproducible paper"

The end

## 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 reproducibility

## 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 reproducibility

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 reproducibility

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 reproducibility

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)

## User-facing software stack definitions

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"
        ...
```

## User-facing software stack definitions

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"
        ...
```

## For stack developers: DSL 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
  ...
```

## Temporary internal representation in Hashdist

```
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]
)
```

## For Hashdist developers

Feed it through a Python pipeline:

```
@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']
        ])
```

## Generated, read by Hashdist developers while debugging

```
{ "import" : [{ "id" : "gcc/apyicmxgafb564zz7rwhwvon7padvxdx"},
               { "id" : "virtual:unix"},
               { "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"]]
}
```