Optimized & Reproducible HPC Software Deployment

... with GNU Guix and free software

Pjotr Prins[1] & Ludovic Courtès[2]

FOSDEM, February 2017

[1] UMC Utrecht, UTHSC GeneNetwork.org
[2] Inria

Recipe for a contemporary HPC cluster environment.

#1. Start with an old & inflexible distro.

#2. Add a layer of home-made “modules”.

Here is an example of loading a module on a Linux machine under bash.

```
% module load gcc/3.1.1
% which gcc
/usr/local/gcc/3.1.1/linux/bin/gcc
```

Now we'll switch to a different version of the module

```
% module switch gcc gcc/3.2.0
% which gcc
/usr/local/gcc/3.2.0/linux/bin/gcc
```

#2b. Tweak the modules.

#2b. Tweak the modules.
#2c. Oh, run-time linker error!

#2b. Tweak the modules.
#2c. Oh, run-time linker error!
#2d. Tweak build flags for user A.

#2b. Tweak the modules.
#2c. Oh, run-time linker error!
#2d. Tweak build flags for user A.
#2e. New versions are out, rebuild!

#2b. Tweak the modules.
#2c. Oh, run-time linker error!
#2d. Tweak build flags for user A.
#2e. New versions are out, rebuild!
#2f. User B unhappy cuz we upgraded.
Ignore?

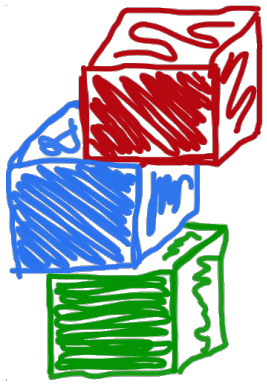
#2b. Tweak the modules.
#2c. Oh, run-time linker error!
#2d. Tweak build flags for user A.
#2e. New versions are out, rebuild!
#2f. User B unhappy cuz we upgraded.
Ignore?
...

#3. Spice up with user-built software!

Application-level package managers [edit]

- Anaconda - a package manager for Python
- Assembly - a partially compiled code library for use in Common Language Infrastructure (CLI) deployment, versioning and security.
- Biicode - a file-focused dependency manager for C/C++ languages and platforms (PC, Raspberry Pi, Arduino).
- Bower - a package manager for the web.
- UPT - a fork of Bower that aims to be a universal package manager, for multiple evironments and unlimited kind of package
- Cabal - a programming library and package manager for Haskell
- Cargo - a package manager for Rust (programming language)
- CocoaPods - Dependency Manager for Objective-C and RubyMotion projects
- Composer - Dependency Manager for PHP
- CPAN - a programming library and package manager for Perl
- CRAN - a programming library and package manager for R
- CTAN - a package manager for TeX
- DUB - a package manager for D

Fixing HPC cluster environments.



easybuild

Spack

Open **boegel** opened this issue on Nov 5, 2013 · 0 comments

boegel commented on Nov 5, 2013 Member

It *seems* like the GCC libraries (e.g. `libiberty.a`) sometimes end up being built with `-fPIC` (e.g. on SL5), and sometimes not (e.g. on SL6), while `eb` is performing the exact same build procedure.

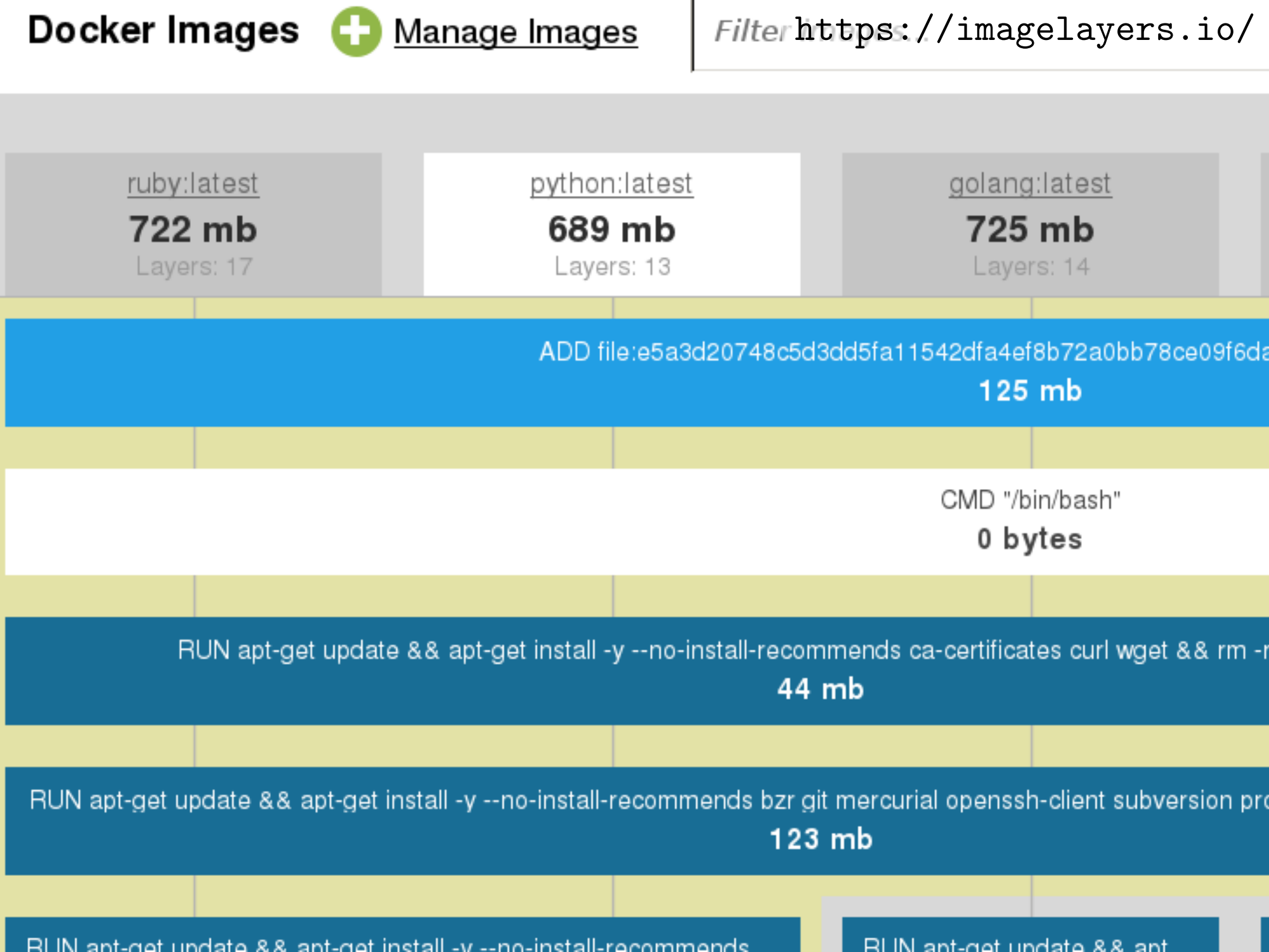
This causes problems for `cairo` (see) and `ExtraE` (part of UNITE), which require `libiberty.a` to be built with `-fPIC`. The `cairo` builds works fine on SL5, but doesn't work on `SL6` (see also hpcugent/easybuild-easyconfigs#494 (comment)).

citibeth commented on Oct 23, 2016 Collaborator

Good news, I ran into this problem too. But only on SOME computers... I don't yet know why some but not all. Anyway... look in the generated `spconfig.py` files, I see the following:

```
env['PATH'] = ":".join(cmdlist("""
    /gpfsm/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/cmake-3.6.1-xfzr
    /gpfsm/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/python-3.5.2-d5i
    /gpfsm/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/netcdf-cxx4-4.3.
    /gpfsm/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/py-numpy-1.11.1-
    /gpfsm/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/udunits2-2.2.20-
    /gpfsm/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/proj-4.9.2-f6543
    /gpfsm/dnb53/rpfische/spack3/lib/spack/env
    /gpfsm/dnb53/rpfische/spack3/lib/spack/env/case-insensitive
    /gpfsm/dnb53/rpfische/spack3/lib/spack/env/gcc
    /gpfsm/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/binutils-2.27-vd
    /home/rpfische/git/modele-control/bin
    /usr/local/other/SLES11.3/openmpi/1.10.1/gcc-5.3/bin
    /usr/local/other/SLES11.3/gcc/5.3.0/bin
    /usr/local/other/SLES11.3/git/2.7.4/libexec/git-core
    /usr/local/other/SLES11.3/git/2.7.4/bin
```

Approach #2:
Give up on packaging.


Docker Images
Manage Images
Filter
https://imagelayers.io/
ruby:latest
722 mb
Layers: 17
python:latest
689 mb
Layers: 13
golang:latest
725 mb
Layers: 14
ADD file:e5a3d20748c5d3dd5fa11542dfa4ef8b72a0bb78ce09f6da
125 mb
CMD "/bin/bash"
0 bytes
RUN apt-get update && apt-get install -y --no-install-recommends ca-certificates curl wget && rm -r
44 mb
RUN apt-get update && apt-get install -y --no-install-recommends bzr git mercurial openssh-client subversion pro
123 mb
RUN apt-get update && apt-get install -y --no-install-recommends
RUN apt-get update && apt

October 20, 2016

Container App ‘Singularity’ Eases Scientific Computing

Tiffany Trader

HPC container platform Singularity is just six months out from its 1.0 release but already is making inroads across the HPC research landscape. It’s in use at Lawrence Berkeley National Laboratory (LBNL), where Singularity founder Gregory Kurtzer has worked in the High Performance Computing Services (HPCS) group for 16 years, and it’s going into other leading HPC centers, including the Texas Advanced Computing Center (TACC), the San Diego Supercomputing Center (SDSC) and many more sites, large and small.


Can we eat it too?


Guix

1. transactional package manager
2. software environment manager
3. APIs & tools to customize environments
4. packaging tools

- started in 2012
- **4,800+ packages**, all free software
- **4 architectures**:
  x86_64, i686, ARMv7, mips64el
- binaries at `https://hydra.gnu.org`
- 0.12.0 released in December 2016

30 Day Summary

Oct 9 2016 — Nov 8 2016

639 Commits

40 Contributors

including 5 new contributors

12 Month Summary

Nov 8 2015 — Nov 8 2016

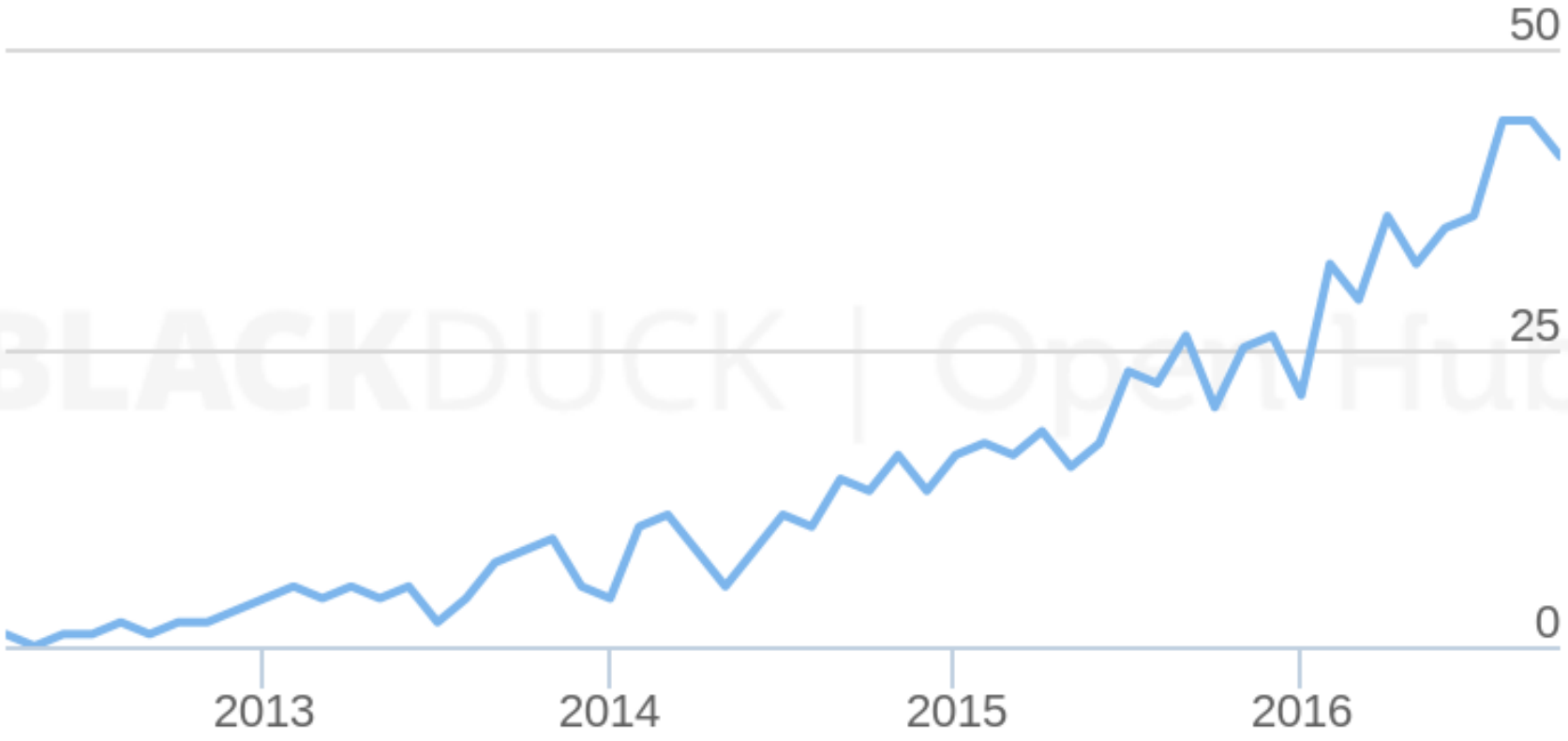
6533 Commits

Up + 1455 (28%) from previous 12 months

106 Contributors

Up + 53 (100%) from previous 12 months

Contributors per Month

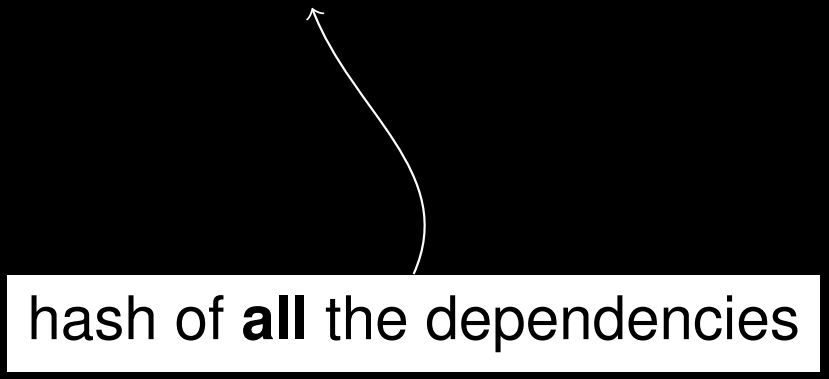


```
$ guix package -i gcc-toolchain openmpi hwloc
…

$ eval `guix package --search-paths`
…

$ guix package --manifest=my-software.scm
…
```

```
$ guix build hello
```

isolated build: chroot, separate name spaces, etc.

```
$ guix build hello
/gnu/store/ h2g4sf72... -hwloc-1.11.2
```



```
$ guix build hello
/gnu/store/ h2g4sf72... -hwloc-1.11.2

$ guix gc --references /gnu/store/...-hwloc-1.11.2
/gnu/store/...-glibc-2.24
/gnu/store/...-gcc-4.9.3-lib
/gnu/store/...-hwloc-1.11.2
```

```
$ guix build hello
/gnu/store/ h2g4sf72... -hwloc-1.11.2

$ guix gc --references /gnu/store/...-hwloc-1.11.2
/gnu/store/...-glibc-2.24
/gnu/store/...-gcc-4.9.3-lib
/gnu/store/...-hwloc-1.11.2
```

(nearly) bit-identical for everyone

Reproducible and User-Controlled Software Environments in HPC with Guix

Ludovic Courtès[1] and Ricardo Wurmus[2]

[1] Inria, Bordeaux, France
[2] Max Delbrück Center for Molecular Medicine, Berlin, Germany

Abstract. Support teams of high-performance computing (HPC) systems often find themselves between a rock and a hard place: on one hand, they understandably administrate these large systems in a conservative way, but on the other hand, they try to satisfy their users by deploying up-to-date tool chains as well as libraries and scientific software. HPC system users often have no guarantee that they will be able to reproduce results at a later point in time, even on the same system—software may have been upgraded, removed, or recompiled under their feet, and they have little hope of being able to reproduce the same software environment elsewhere. We present GNU Guix and the functional package management paradigm and show how it can improve reproducibility and sharing among researchers with representative use cases.

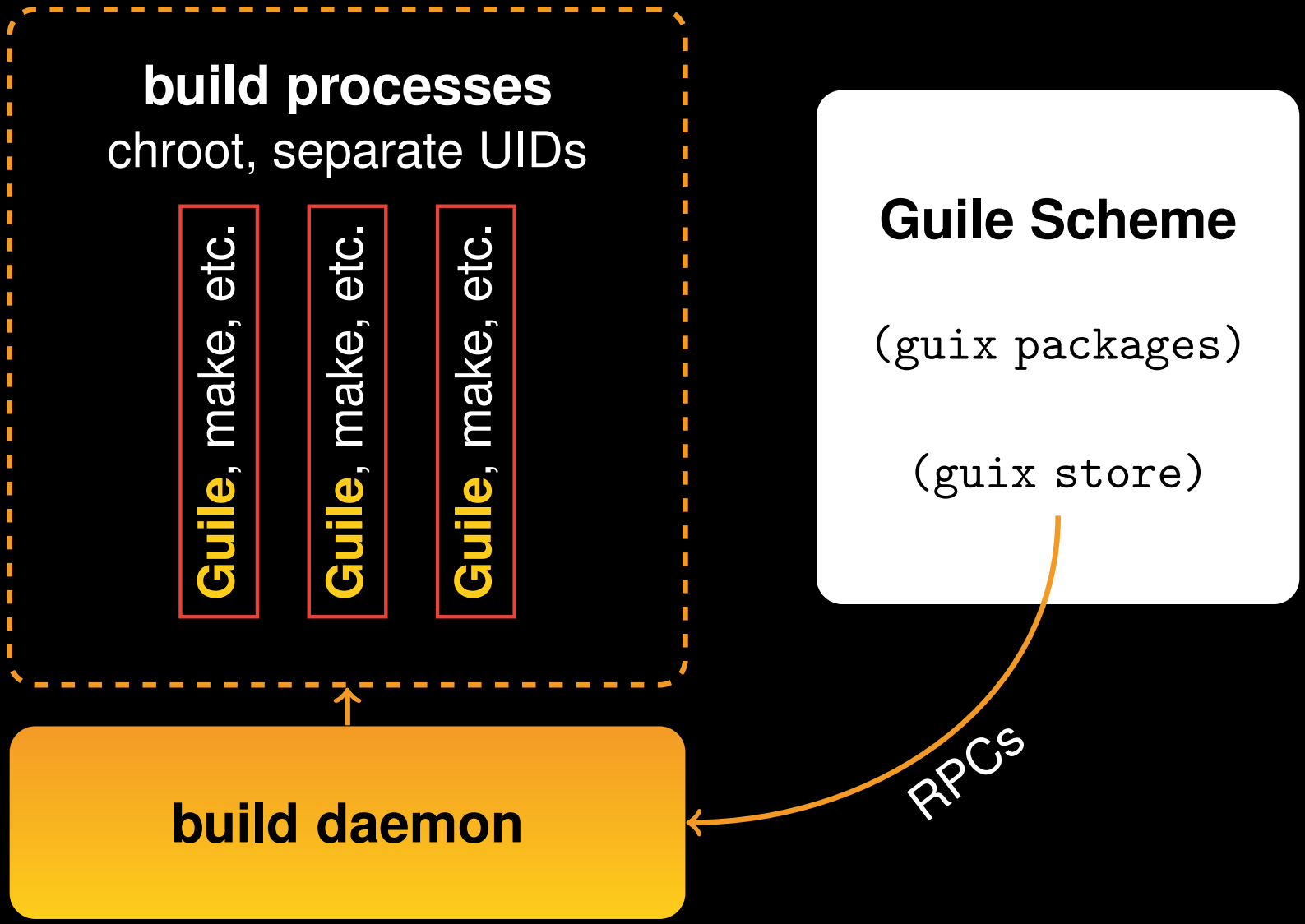
Creating package variants at the command line

```
$ guix build hwloc \
    --with-source=./hwloc-42.0rc1.tar.gz
...
```

```
$ guix build hwloc \
    --with-source=./hwloc-42.0rc1.tar.gz
...


$ guix package -i mumps \
     --with-input=scotch=pt-scotch
...
```

Your personal packages or variants in `GUIX_PACKAGE_PATH`!

HPC & non-root usage.


build processes
chroot, separate UIDs
Guile, make, etc.
Guile, make, etc.
Guile, make, etc.
Guile Scheme
(guix packages)
(guix store)
RPCs
build daemon

Allowing for non-root usage

0. run build daemon **as non-root**
1. rely on Linux “**user namespaces**”
2. make binaries **relocatable**

Allowing for non-root usage

0. run build daemon **as non-root**
   - not reproducible
   - prevents use of pre-built binaries
1. rely on Linux “**user namespaces**”

2. make binaries **relocatable**

Allowing for non-root usage

0. run build daemon **as non-root**
   - not reproducible
   - prevents use of pre-built binaries
1. rely on Linux “**user namespaces**”
   - awesome!
   - ... but support is missing on some systems
2. make binaries **relocatable**

Relocatable binaries.

Insight 1

The first key insight: Guix store paths provide unique fingerprints:

`/gnu/store/m9vxvhdj691bq1f85lpflvnhcvrdilih-glibc-2.23/lib/libc.`

Linked libraries

ldd ‘which ldc2‘

```
libconfig.so.9  /gnu/store/1v4an...-libconfig-1.5/lib/libconfig.
librt.so.1      /gnu/store/m9vxv...-glibc-2.23/lib/librt.so.1
libdl.so.2      /gnu/store/m9vxv...-glibc-2.23/lib/libdl.so.2
libpthread.so.0 /gnu/store/m9vxv...-glibc-2.23/lib/libpthread.so
libz.so.1       /gnu/store/5992i...-zlib-1.2.8/lib/libz.so.1
libm.so.6       /gnu/store/m9vxv...-glibc-2.23/lib/libm.so.6
libstdc++.so.6  /gnu/store/9nifw...-gcc-4.9.3-lib/lib/libstdc++.
libgcc_s.so.1   /gnu/store/9nifw...-gcc-4.9.3-lib/lib/libgcc_s.s
libc.so.6       /gnu/store/m9vxv...-glibc-2.23/lib/libc.so.6
                /gnu/store/m9vxv...-glibc-2.23/lib/ld-linux-x86-
```

Relocate

- Guix binaries have unique fingerprints for PATHs
- Replace these with the target prefix
- Only dependency is the kernel

After relocation

ldd ~/opt/ldc-test/ldc-1.1.0-pk9rkm4zvdp6pglam7s2/bin/ldc2

```
qlibconfig.so.9  ~/opt/ldc-test/libconfig-1.5-1v4anv1.../lib/lib
librt.so.1       ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/librt.so
libdl.so.2       ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/libdl.so
libpthread.so.0 ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/libpthre
libz.so.1        ~/opt/ldc-test/zlib-1.2.8-5992iq1.../lib/libz.so
libm.so.6        ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/libm.so.
libstdc++.so.6  ~/opt/ldc-test/gcc-4.9.3-lib-9nifwk7.../lib/libs
libgcc_s.so.1   ~/opt/ldc-test/gcc-4.9.3-lib-9nifwk7.../lib/libg
libc.so.6        ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/libc.so.
                 ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/ld-linux
```

What really happened here?

- All Guix packages are isolated in the store
- Path is a fingerprint, e.g. `/gnu/store/m9vxvhdj691bq1f85lpflvnhcvrdilih-glibc-2.23`
- Scan all files and replace fingerprints with relative path `~/opt/ldc-test/glibc-2.23-m9vxvhdj691bq1f85lpf`
- First attempt by using Eelco Dolstra's Patchelf tool worked for shared libs by rewriting `RPATH` in binaries

Other files

- Text files that reference the store can be rewritten (Ruby, Perl, bash scripts)
- Some formats are not zero-terminated (compiled Python and JVM files)
- Also in ELF files there are references that are not zero-terminated
- Solution: keep the file path at exactly the same length and patch all

Insight 2

The second key insight: if a path gets rewritten with the exact same size string it will always work (unless there is encryption or some CRC checking)

```
/gnu/store/m9vxvhdj691bq1f85lpflvnhcvrdilih-glibc-2.23/lib/libc.
/home/user/opt/ldc-test/glibc-2.23-m9vxvhdj691bq1f851p/lib/libc.
```

Same size patching

1. start from store path, e.g.
   `/gnu/store/m9vxvhdj691bq1f85lpflvnhcvrdilih-glibc-2.23`
2. reverse the contents
   `/gnu/store/glibc-2.23-m9vxvhdj691bq1f85lpflvnhcvrdilih`
3. overwrite with prefix and shorten the HASH value to match the same size
   `/home/user/opt/ldc-test/glibc-2.23-m9vxvhdj691bq1f851p`
4. and replace in all files

Example

So, store path

```
/gnu/store/m9vxvhdj691bq1f85lpflvnhcvrdilih-glibc-2.23/bin/l
```

```
prefix /home/usr/opt/ldc-test/ becomes
/home/user/opt/ldc-test/glibc-2.23-m9vxvhdj691bq1f851p/bin/l
```

```
prefix /usr/local/share/ldc-1.1.0/ becomes
/usr/local/share/ldc-1.1.0/glibc-2.23-m9vxvhdj691bq1f8/bin/l
```

Note: prefix can be up to $\sim$ 40 letters long

So far...

Successfully compiled and run

- ldc2 1.1.0: the LLVM D compiler
- ruby 2.3.0: with ssl and nokogiri
- sambamba: tool used in many sequencing HPCs around the world
- more to come, including OpenCL, R, Python and Julia

Cross compile

- Install compilers that can cross compile binaries
- LLVM can output C code
- Provide GNU Guix packages for Intel PHI and NVIDIA TESLA
- GNU Guix has elegant support for different targets, including a build farm for ARM, ...

Wrap-up.

Future

Implications carry beyond HPC

- Automated builds with testing for different architectures
- Repository of binary packages
- One-click installs: download and run `install.sh`
- Ship software easily
- Talking about a holy grail…

Summary

- Guix supports **reproducible software environments**
- ... allows for **experimentation** through customization
- relocation allows **unprivileged** Guix usage in HPC

Acknowledgements

- Roel Janssen (@roelj), Dennis Mungai (@Brainiarc7), and Frederick Muriithi (@fredmanglis) for helping with packaging D compilers, sambamba, Ruby packages, OpenCL etc.
- The GNU and GNU Guix communities (many, many talented individuals)

GuixSD

https://gnu.org/software/guix/