Reproducible packaging and distribution of software with GNU Guix

Distributing software with Guix

Pjotr Prins

FOSDEM

February 5th, 2017

UMC Utrecht/UTHSC GeneNetwork.org

Holy grail

- Controlled software deployment
- Control the full dependency graph
- Write once and reproduce *any time*

This talk

1. Deployment in a development/testing/staging/production environment
2. Easy installation
3. Distributed workflows
4. Orchestration of services

Deployment (1)

- The GeneNetwork service, essentially simple, represents a complex set of dependencies
- Redid, MySQL, nginx and related
- Python-scipy, jinja, werkzeug, sqlalchemy, PIL
- R/qtl, R-ctl, R-wgcna
- gemma, pylmm, CTL, plink
- Openblas, Atlas, Ruby(?) And growing!

`http://biogems.info/contrib/genenetwork/genenetwork2-2.0-a8fcff4.svg`

Deployment

- Development, testing, staging, production
- When we started it was a mess
- Every stage was managed ad-hoc
- No chance of reproducibility
- Started packaging using a checked out GNU Guix git tree with antother git tree GUIX_PACKAGE_PATH (100+ packages, some duplicates)

`https://github.com/genenetwork/guix-bioinformatics/tree/master/gn/packages`

Shared profiles

- To manage development/testing/staging/production we now use shared profiles:
- /usr/local/shared/gn-staging/genenetwork2-2.0-a8fcff4
- For each version main Guix package tree is fixed in time with GUIX_PACKAGE_PATH tree using git
- Guix channels would be an improvement

Channels

- GUIX_PACKAGE_PATH works, but is also a disconnect
- Channels would make it easier to fixate versioned trees
- Different versions of software would become channels (say an ancient Python)
- Channels would also make it easier to merge patches with Guix git master
- All stuff I now handle by juggling git branches (ouch) and can't explain to others

Easy installation (2)

- Docker, Brew, Conda, Easybuild. . .
- What is needed is user level easy installation
- Without people having to install Guix daemon and paths
- We now have ‘relocatable’ Guix

Insight 1

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

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

Linked libraries

ldd ‘which ldc2‘

```
libconfig.so.9  /gnu/store/1v4anv1...-libconfig-1.5/lib/libconfig.so.9
librt.so.1      /gnu/store/m9vxvh...-glibc-2.23/lib/librt.so.1
libdl.so.2      /gnu/store/m9vxvh-glibc-2.23/lib/libdl.so.2
libpthread.so.0 /gnu/store/m9vxvh...-glibc-2.23/lib/libpthread.so.0
libz.so.1       /gnu/store/5992iq1...-zlib-1.2.8/lib/libz.so.1
libm.so.6       /gnu/store/m9vxvhd...-glibc-2.23/lib/libm.so.6
libstdc++.so.6  /gnu/store/9nifwk7...-gcc-4.9.3-lib/lib/libstdc++.so.6
libgcc_s.so.1   /gnu/store/9nifwk7...-gcc-4.9.3-lib/lib/libgcc_s.so.1
libc.so.6       /gnu/store/m9vxvh...-glibc-2.23/lib/libc.so.6
                /gnu/store/m9vxvh...-glibc-2.23/lib/ld-linux-x86-64.so.2
```

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/libconfig.so.9
librt.so.1       ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/librt.so.1
libdl.so.2       ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/libdl.so.2
libpthread.so.0 ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/libpthread.so.0
libz.so.1        ~/opt/ldc-test/zlib-1.2.8-5992iq1.../lib/libz.so.1
libm.so.6        ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/libm.so.6
libstdc++.so.6   ~/opt/ldc-test/gcc-4.9.3-lib-9nifwk7.../lib/libstdc++.so.6
libgcc_s.so.1    ~/opt/ldc-test/gcc-4.9.3-lib-9nifwk7.../lib/libgcc_s.so.1
libc.so.6        ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/libc.so.6
                 ~/opt/ldc-test/glibc-2.23-m9vxvh.../lib/ld-linux-x86-64.so.2
```

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

`http://nixos.org/patchelf.html`

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.so
/home/user/opt/ldc-test/glibc-2.23-m9vxvhdj691bq1f851p/lib/libc.so
```

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

`https://github.com/pjotrp/guix-relocate`

Example

So, store path

```
/gnu/store/m9vxvhdj691bq1f85lpflvnhcvrdilih-glibc-2.23/bin/ldc2

prefix /home/usr/opt/ldc-test/ becomes
/home/user/opt/ldc-test/glibc-2.23-m9vxvhdj691bq1f851p/bin/ldc2

prefix /usr/local/share/ldc-1.1.0/ becomes
/usr/local/share/ldc-1.1.0/glibc-2.23-m9vxvhdj691bq1f8/bin/ldc2
```

Note: prefix can be up to ~ 40 letters long - or longer

Example

Download (42Mb), unpack (137Mb), install (3 sec.) and run Ldc the latest LLVM based D compiler 1.1.0:

```
wget http://biogems.info/contrib/genenetwork/ \
    pk9rkm4zvdp6pglam7s280x1x8y5rvbz-ldc-1.1.0-x86_64.tar.bz2
  (md5sum fe2508135eadc87fcc31027524c11ec5)
time ./install.sh TARGETDIR
TARGETDIR/ldc*/bin/ldc2 --version
```

https://forum.dlang.org/post/ckoiqzoatxyjlpakufsa@forum.dlang.org

Success!

Successfully built on Guix and packaged as relocatable binary:

- 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

`https://github.com/pjotrp/guix-relocatable-binary-packages`

Binary downloads

- Created for HPC environments, but wider application is clear
- Ship complicated dependencies
- 1-click installs are a possibility (script unpacking tar)
- Website for relocatable binary downloads - what about security?

Workflows (3)

- In biomedical the common workflow language (CWL) has momentum
- Started as a descriptive document
- But has lost determinism
- Not least because Javascript is now part of the specification
- Insight: workflow ‘standards’ become very complicated quickly

Workflows in Guix

- Even so, workflows are essentially simple (key insight):
  - Serial execution
  - Parallel execution (scatter gather)
  - Optimizations (order)
- Guix provides determinism and serial out of the box
- Parallel execution is part of the build farm

Also Roel Janssen's talk on workflows @15:30 in this room

Orchestration (4)

- Many companies face issues with orchestration
- Orchestration of services is complicated
- Run on different servers
- Launch order matters
- Guix provides services as a single machine ‘orchestration’
- Perhaps using workflows should look at expanding on that idea for multiple machines

Conclusion

1. GNU Guix allows for controlled and sane software deployment with profiles and git (and soon channels)
2. GNU Guix relocatable binary packages do not require administrative privileges and can become 1-click installs
3. GNU Guix will handle workflows
4. Orchestration should be on the agenda

Acknowledgements

- Roel Janssen (@roelj), for working on a new workflow engine
- GNU Guix project leaders Ludovic Courtès and Ricardo Wurmus
- The GNU and GNU Guix communities (many, many talented individuals)
- Prof. Robert W. Williams and the GeneNetwork.org project