FOSDEM 2018

HPC, Big Data & Data Science devroom

Feb 4th 2018, Brussels (Belgium)

Installing software for scientists on a multi-user HPC system

A comparison between:

CONDA easybuild GuixHPC Nix Spack

Kenneth Hoste

kenneth.hoste@ugent.be

GitHub: @boegel

Twitter: @kehoste

Installing software for scientists on a multi-user HPC system

- getting scientific software installed can be challenging
    - lack of documentation & good software engineering practices
    - non-standard installation procedures
    - dependency hell
    - ...

> " *If we would know what we are doing, it wouldn't be called 'research'.* "

- scientists mostly care about the *science*
    - they're often not software engineers or system administrators
    - the software they need should be (made) easily accessible

Installing software for scientists
on a multi-user HPC system

- a supercomputer is very different from your laptop...
  - both in good (performance, parallellism) and 'bad' ways (ease of use)
- often broad spectrum of users, with varying requirements
  - typically central installation of (scientific) software
  - multiple software versions (& variants) side-by-side
  - software installations remain available 'indefinitely'
- performance is key
  - get the most out of the available hardware (processor architecture, ...)
  - maximise amount of "science" that can be done
  - 10% performance difference can be a big deal...

Disclaimer & acknowledgements

- my intention is to make this an *objective* comparison
- not easy as lead developer/release manager of easybuild
- I spent hours of hands-on with each of the tools to familiarise myself
- there is definitely still some personal bias here and there...
- thanks to many people for their feedback:
    - Todd Gamblin
    - Ludovic Courtès
    - Ricardo Wurmus
    - Valentin Reis
    - Bruno Bzeznik
    - Ward Poelmans
    - Jillian Rowe
    - + anyone else who answered any of my questions...

30-second introductions

CONDA

Spack

easybuild

Nix

GuixHPC

- OS: Linux, macOS, Windows
- impl. in Python (2.7 or >= 3.3)
- **target audience: end users, scientists**

CONDA

https://conda.io

- focus:
  - binary packages
  - quick & easy software installation
  - cross-platform

package, dependency and environment management "for any language"

(originally created for Python, but now also supports C, C++, FORTRAN, R, ...)

- tool for installing binary packages and setting up 'environments'
- included in Anaconda: optimised Python/R distribution (batteries incl.)
- packages are available via *Anaconda cloud* and many other '*channels*'
- package recipes are written in YAML syntax + a script (.sh or .bat)
  - building of packages is done via "`conda build`"
  - GitHub organisation for hosting recipes: https://conda-forge.org
- supported software: > 3,500

- OS: Linux, Cray, (macOS)
- impl. in Python (2.6 or 2.7)
- **target audience: HPC user support teams**

easybuild

http://easybuilders.github.io/easybuild

- focus:
  - building from source
  - easy installation of software
  - good performance

framework for building & installing (scientific) software on HPC systems

- build procedures are implemented in *easyblocks* (Python modules), which leverage the functionality of the EasyBuild *framework*
- separate *easyconfig* files specify (in Python syntax) what to install, and using which *toolchain* (compiler + MPI/BLAS/LAPACK/FFT libraries)
- aims for good performance by default: compiler options, libraries, ...
- generates environment module files (easy interface for end users)
- various features to allow site-specific customisations
  - support for using own easyconfig files (recipes), 'plugins', hooks, ...
- supported software: > 2,000 (> 1,300 + > 700 'extensions')

- OS: Linux, macOS, Unix
- implemented in C++
- **target audience: system administrators, (experienced) end users, ...**

Nix

https://nixos.org/nix

- focus:
  - binary installations
  - isolated build environment
  - portability

the purely functional package manager

- package (and configuration) manager for NixOS, but can also be used stand-alone on other Unix systems
- strong focus on (bitwise) reproducibility through build isolation, etc.
- supports atomic package upgrades & rollbacks
- downloads and installs binary packages (or builds from source if not available)
- multi-user support via profiles with `nix-env`
- package recipes are implemented in custom Nix DSL
- supported software: > 13,000 (+ 12,000 Haskell packages)

- OS: GNU/Linux
- implemented in Scheme, C++
- **target audience: system administrators, (experienced) end users, ...**

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

- focus:
  - binary installations
  - isolated build environment
  - free software & GNU philosophy

the GNU package manager

- package manager for GuixSD, the Guix System Distribution (+ GNU Hurd), but can also be used on other GNU/Linux distributions
- design is quite similar to Nix, but different implementation
  - package definitions in GNU Guile (Scheme) rather than custom Nix DSL
  - Guix can leverage the Nix build daemon if available
- also strong focus on (bitwise) reproducibility of installations
- only supports free software, no proprietary software
- transactional upgrades & rollbacks, per-user profiles, etc.
- supported software: > 6,500

- OS: Linux, macOS, Cray
- impl. in Python (>=2.6 or >=3.3)
- **target audience:**
  (scientific) software developers

Spack

https://spack.io

- focus:
  - building from source
  - flexibility
  - cross-platform

Spack is a flexible package manager
for supercomputers, Linux, and macOS

- supports multiple (software) versions, configurations, compilers, ...
- quite similar to EasyBuild in some ways, but has a different design & focus
  - packages are (also) Python modules, but no separate 'recipe' files (cfr. easyconfigs)
  - in-memory DAG resolution, dependency resolution, database of installed packages
  - support for exposing installations through environment modules (or dotkit)
- powerful CLI to specify partial DAG w.r.t. dependencies, compiler, etc.

```
spack install mpileaks@1.1.2 %gcc@4.7.3 +debug ^libelf@0.8.12
```

- supported software: > 2,300

Project comparison

	CONDA	easybuild	GuixHPC	Nix	Spack
platforms	Linux, macOS, Windows	Linux, Cray	GNU/Linux	Linux, macOS, Unix	Linux, macOS, Cray
implementation	Python 2/3, YAML	Python 2	Scheme, Guile	C++, Nix (DSL)	Python 2/3
supp. software	> 3,500	> 2,000	< 6,500	> 13,000	> 2,300
releases, install & update					
documentation					
configuration					
usage					
time to result					
performance					
reproducibility					

this comparison table will be completed in the remainder of this talk with *stars*

★★★★ excellent

★★★½ very good

★★★ good

★★½ ok

★★ average

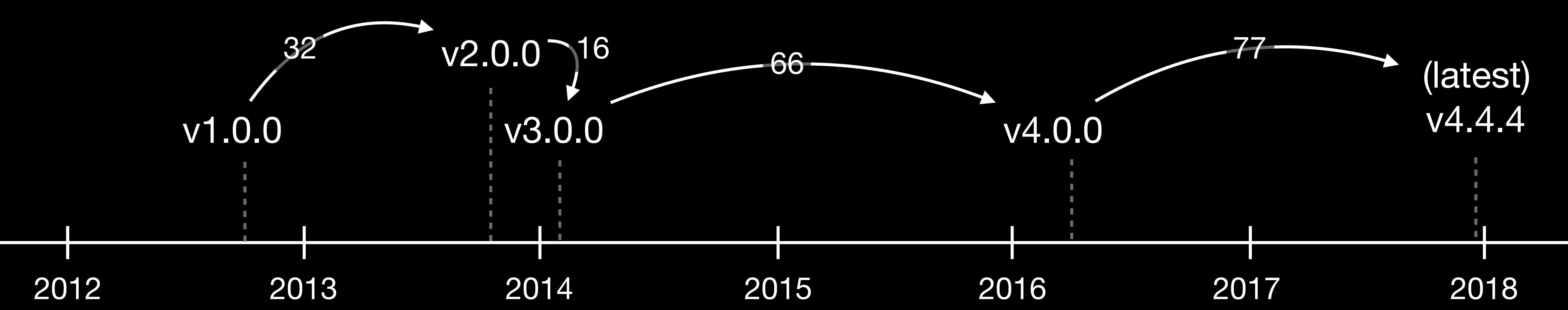
★ bad

Releases, installing & updating

CONDA

sudo required?
- installation: no
- usage: no

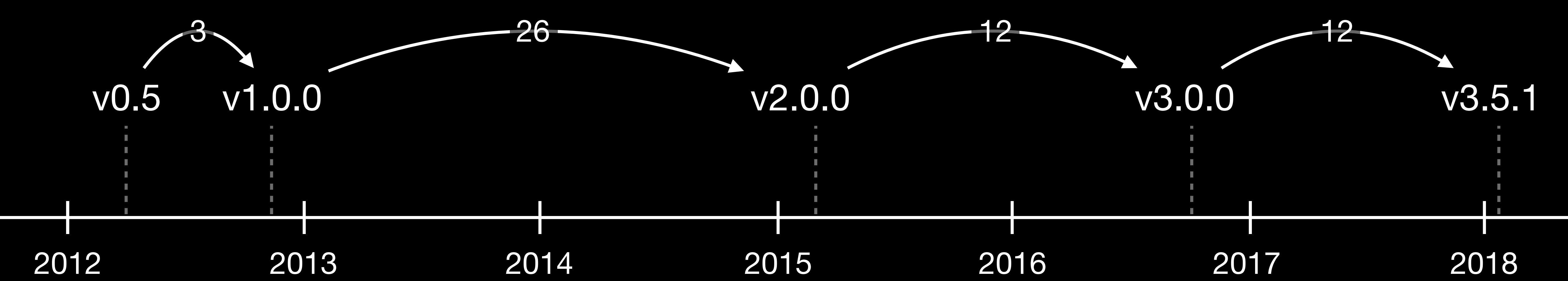
- **195** releases since September 2012
- install via shell install script (via *Miniconda* or *Anaconda*)
- self-update using "`conda update conda`"
- dependencies: *none* (even Python is included in installation!)



Releases, installing & updating

sudo required?
- installation: no
- usage: no!

- 58 releases since April 2012
  - stable (v1.0) since November 2012
  - 3 years of in-house development prior to first public release
- installation via custom bootstrap script, or standard Python tools (`pip`, ...)
- self-update using "`eb --install-latest-eb-release`"
- dependencies: environment modules, Python 2.x, setuptools, C++ compiler, ...

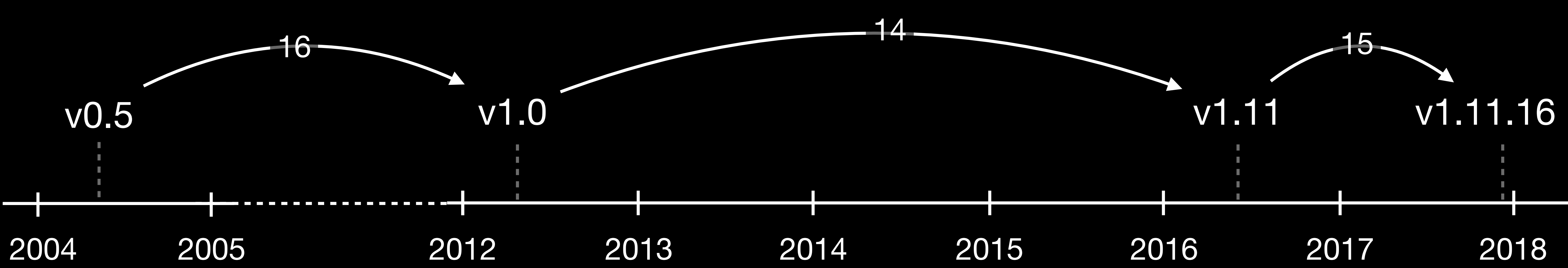


Releases, installing & updating

Nix

sudo required?
- installation: *yes*
- usage: no

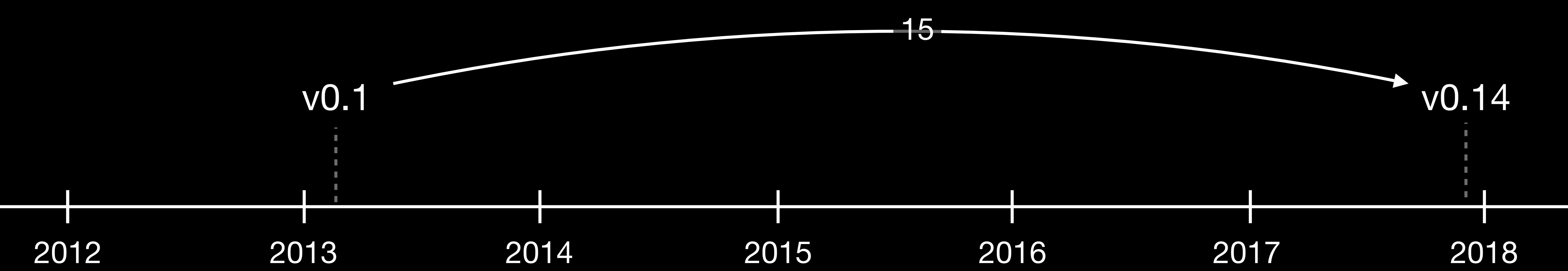
- 49 releases since April **2004**
- stable (v1.0) since May 2012
- custom install script to install binary release (or from build source)
- build daemon (`nix-daemon`) required (as `root`) for multi-user support
- self-update via `nix-channel --update && nix-env --install nix`
- dependencies: none (unless you build Nix from source)



Releases, installing & updating

- 17 releases since January 2013
- still in *beta* (no v1.0 yet)
- install by unpacking binary distribution, or build from source
  - no installation script available, manual installation process...
  - build daemon (`guix-daemon`) required (as `root`)
- self-update using "`guix pull`"
- dependencies: Guile, libgcrypt, make, ..

sudo required?
- installation: *yes*
- usage: no



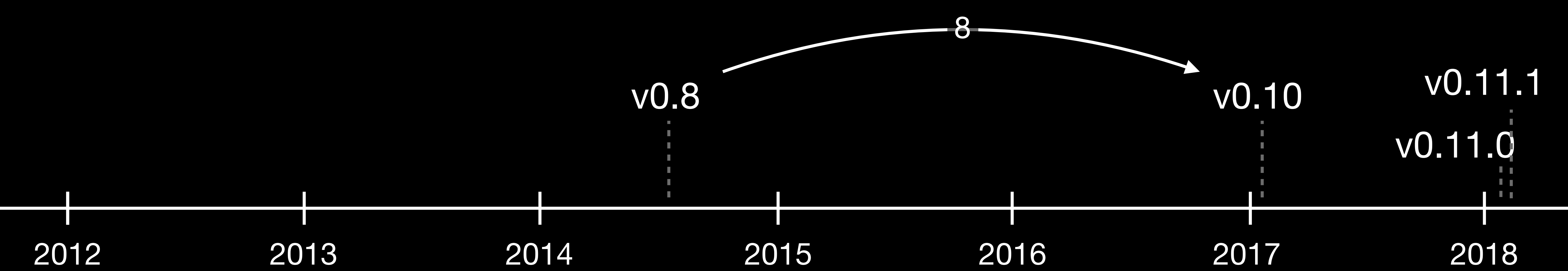
Releases, installing & updating

Spack

sudo required?
- installation: no
- usage: no

- 12 releases since July 2014
- still in *beta* (no v1.0 yet)
- install by unpacking source tarball or using "`git clone`" *(recommended)*

  + setting up environment (update `$PATH` or source a script)
- update Spack using "`git pull`"
- dependencies: C/C++ compiler, git, curl, ...



Documentation

All 5 projects have good to excellent documentation!

(but there's always room for improvement...)

★★★½	CONDA	conda.io
★★★	easybuild	easybuild.readthedocs.io
★★★½	GuixHPC	www.gnu.org/software/guix/manual/guix.html
★★★½	Nix	nixos.org/nix/manual
★★★½	Spack	spack.readthedocs.io

Configuration

★★★½ CONDA

- software installation prefix *can* be specified per conda environment

```
conda create --prefix <path>
```

- default is to install software in `$HOME/.conda`

★★★ easybuild

- some minimal configuration is highly recommended:
  - software/modules installation prefix (default: `$HOME/.local/easybuild`)
  - location of build directories (recommended: `/tmp`, `/dev/shm`, ...)
  - also: modules tool, syntax for module files, ...
- via configuration files, environment or command line options

Configuration

GuixHPC

- limited to no required configuration, except build users for daemon
- software is installed into `/gnu/store` (hard to change)

Nix

- limited to no required configuration, except build users for daemon
- software is installed into `/nix/store` (hard to change)

Spack

- software is installed into `<spack>/opt/spack` (easy to change)
- several optional configuration settings available

Basic usage

★★★

- first, create a conda environment:

```
conda create --prefix $HOME/my_fftw
```

- activate the environment to install (& use) the software:

```
source activate $HOME/my_fftw
```

- installing software into current conda environment:

```
conda install -c conda-forge fftw
```

- to install other software (versions), either:

  i. try to find a conda package for it somewhere (other channel, ...)

  ii. create/update `meta.yaml` (and `build.sh`) and build package yourself

```
conda build recipe
conda install --local recipe
```

Basic usage

- search for available easyconfig files

```
eb --search fftw
```

- install software (+ toolchain/dependencies) by specifying an easyconfig:

```
eb FFTW-3.3.7-gompi-2018a.eb --robot
```

- to use the software, load the corresponding generate module file:

```
module load FFTW/3.3.7-gompi-2018a
```

- to install other software versions (or use another toolchain), either:
  i. find an easyconfig file (+ easyblock, if needed) for it somewhere
  ii. adjust an existing easyconfig file, or use `eb --try-*`
  iii. compose an easyconfig file yourself (+ easyblock for complex software)

Basic usage

Nix ★★★

- searching for available software

```
nix-env -qa 'fftw.*'
```

- installing software (all 3 variants of FFTW, for different precisions)

```
nix-env --install 'fftw.*'
```

- installations are added to your Nix profile by default, so ready to use
- to install other software (versions):
  - customise existing Nix package, then `nix-env --install ...`
  - new Nix package + build script, then `nix-env --install -f ...`

Basic usage

- searching for available software

```
guix package --search fftw
```

- installing software

```
guix package --install=fftw
```

- installations are added to your Guix profile by default, so ready to use

- to install other software (versions):
  - update existing package file, run `guix package -i <software>`
  - define package (in Scheme), run `guix package -f pkg.scm`

Basic usage

Spack

- install software (+ dependencies) with system compilers

```
spack install fftw
```

- install software (+ dependencies) with a particular compiler

```
spack install gcc@6.4.0
spack compiler add opt/spack/spack/linux-*/gcc-6.4.0
spack install fftw %gcc@6.4.0
```

- to use the software, load it:

`spack load fftw` or `spack load fftw %gcc@6.4.0`

- to install other software (versions):
  - `spack install foo@new-version` (if you're lucky)
  - or maybe need update the 'spackage' (`<software>/package.py`)

Time to result

- quick installation when binary packages are used/favoured:

- slower installation when software is built from source
  (but usually fully autonomous)

- EasyBuild requires toolchain to be available (usually also built from source)
  (existing compilers & libraries can be leveraged too if desired)

- Spack picks up system compilers by default

- Spack is also looking into "architecture-aware" binary packaging
  (see Todd's presentation next!)

Time to result: installing FFTW

(using latest release of each tool)

CONDA ★★★★

FFTW 3.3.7
(binary install)
~25 sec.

GuixHPC ★★★★

FFTW 3.3.5
(binary install)
~2.5 min.

Nix ★★★★

FFTW 3.3.7
(binary install)
~10 sec.

easybuild ★★½

FFTW 3.3.7
(from source)

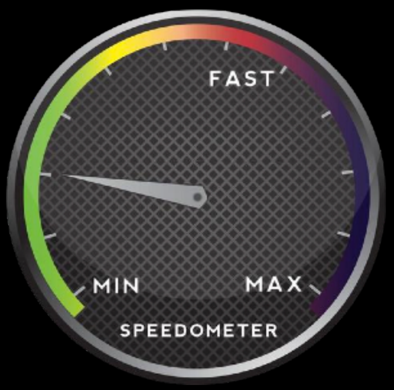
deps (incl. toolchain): ~32 min.
build & install FFTW: ~6 min.
testing: ~32 min. *TOTAL: ~70 min.*

Spack ★★★

FFTW 3.3.6-pl2
(from source)

with system GCC: ~16min. (incl. deps)
with GCC 6.4.0: ~20 min. (incl. deps)
(+ 29 min. to first install GCC 6.4.0)

Performance of installed software



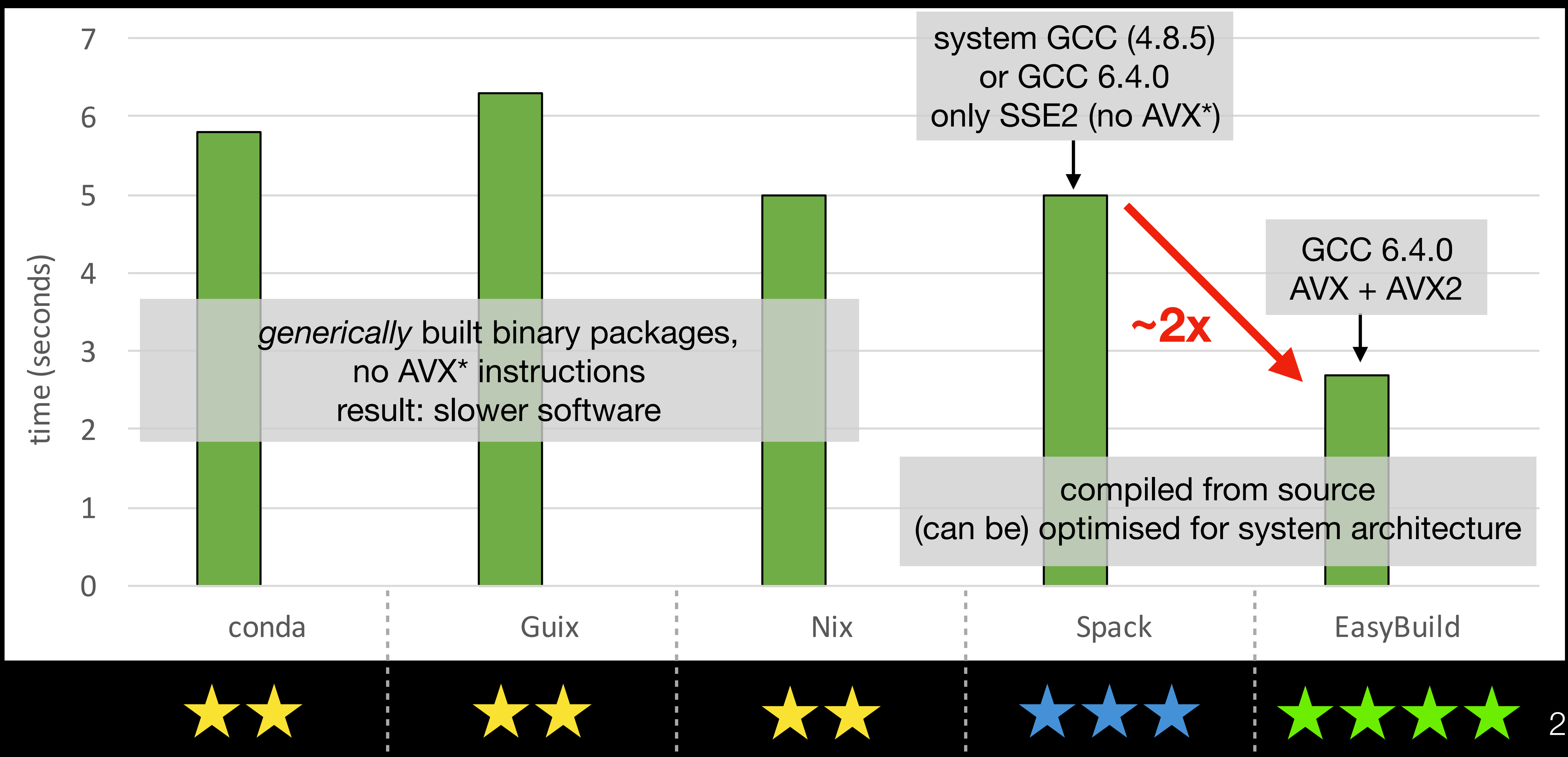
- installing binary packages (usually) implies:
  - installing generically compiled software
  - software installations may not fully exploit system resources
  - sacrificing lower runtime performance for quick installation

- compiling from source *allows* specifically targeting system architecture
  - `gcc -O2 -march=native ...`
  - leverage advanced processor features like AVX2, AVX512, ...
  - trading portability of installations for better runtime performance

- whether you care (much) or not depends heavily on context...
  - quite important on supercomputers!

Performance of FFTW installation

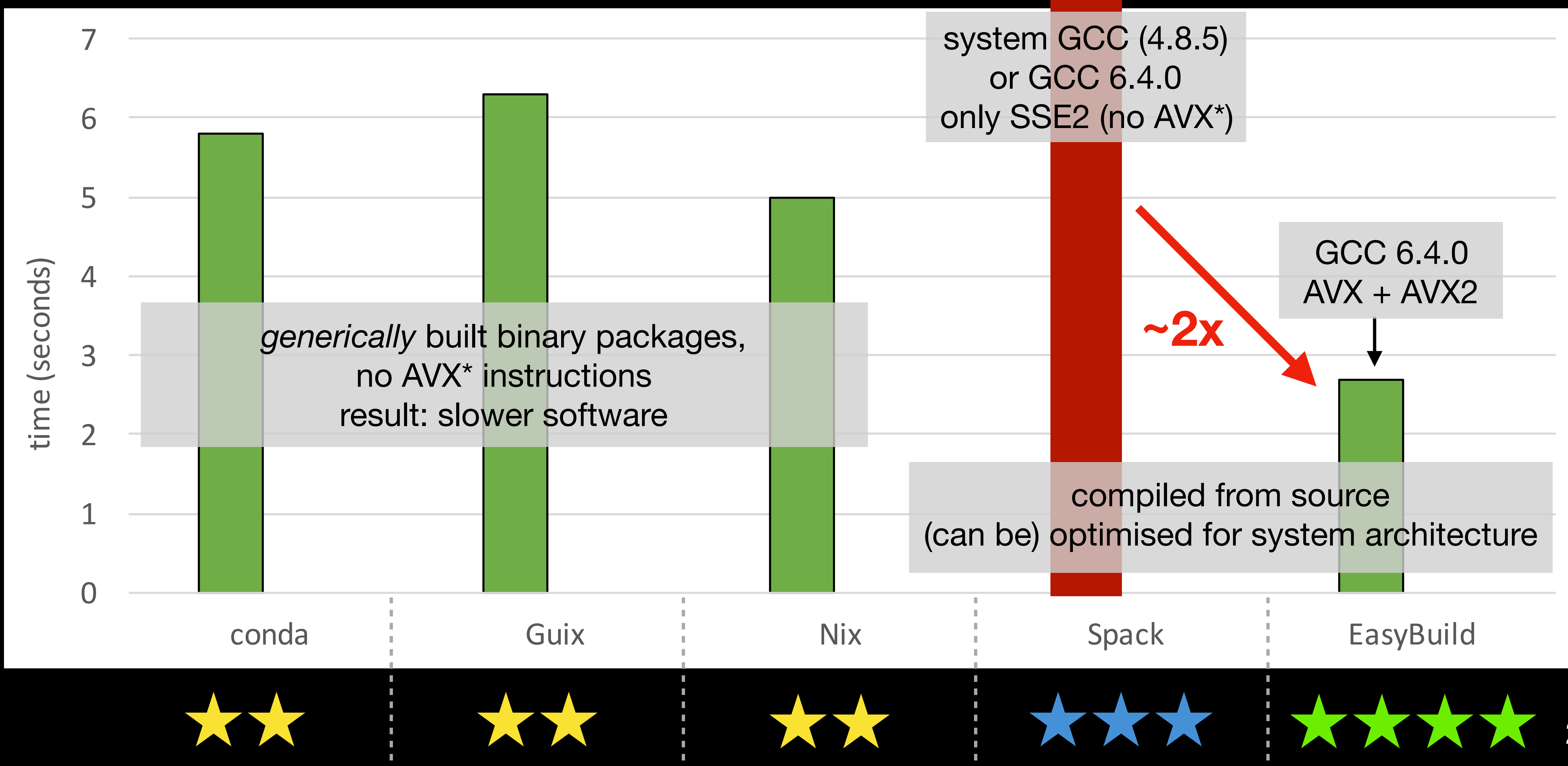
- single-core test from http://micro.stanford.edu/wiki/Install_FFTW3
  - N0, N1 set to 8192 to obtain sufficiently 'long' run times
- **timings are for *default installations* (no tweaking)**
- test system: CentOS 7.4, Intel E5-2680v3 (Haswell-EP) 2.5GHz



Performance of FFTW installation

- single-core test from http://micro.stanford.edu/wiki/Install_FFTW3
  - N0, N1 set to 8192 to obtain sufficiently 'long' run tim[illegible]
- **timings are for *default installations* (no tweaking)**
- test system: CentOS 7.4, Intel E5-2680v3 (Haswell-E[illegible].5GHz

really bad performance with Spack 0.11.0 due to building with -O0 :-/



Other aspects we did not cover

- community
- unit & regression testing
- security
- key features
  - easybuild: support for combining multiple installation prefixes, GitHub integration, distributed software installation, dry run mode, packaging via FPM, support for user-defined hooks, ...
  - GuixHPC, Nix : *bitwise* reproducibility of installations, ...
  - Spack: (very) flexible dependency management, support for binary caching, "virtual" packages (e.g. MPI), variants, ...
- (+ much more...)

And the winner is:

	CONDA	easybuild	GuixHPC	Nix	Spack
platforms	Linux, macOS, Windows	Linux, Cray	GNU/Linux	Linux, macOS, Unix	Linux, macOS, Cray
implementation	Python 2/3, YAML	Python 2	Scheme, Guile	C++, Nix (DSL)	Python 2/3
supp. software	> 3,500	> 2,000	< 6,500	> 13,000	> 2,300
releases, install & update	★★★½	★★★½	★★½	★★½	★★
documentation	★★★½	★★★	★★★½	★★★½	★★★½
configuration	★★★½	★★★	★★	★★	★★★½
usage	★★★	★★★	★★★	★★★	★★★
time to result	★★★★	★★½	★★★★	★★★★	★★★
performance	★★	★★★★	★★	★★	★★★
reproducibility	★★★	★★	★★★★	★★★★	★★

And the winner is: well, it depends...

- ***profile*** of person installing software + profile of end users
  - scientist *vs* software developer *vs* HPC support team *vs* sysadmin
- prior ***experience*** with software installation & compilation
  - can you figure things out if something fails?
- ***use case*** for the software you are installing
  - only to play around with, or for production usage?
  - handful of small experiments, or lots of large-scale calculations?
- whether you are concerned about time to result, reproducibility, security, ...

Linux Windows Mac

FOSDEM'18 talk making waves...

(before it actually happened...)

- **Spack** v0.11.1 bugfix release
    - quickly after v0.11.0 (first Spack release in ~ 1 year)
    - important fix for accidental compilation with -O0
    - problem encountered when testing performance of FFTW install
- easy installation *script* for GuixHPC
    - as reaction to my questions on manual installation procedure
- excellent blog post by Ludovic Courtès on portability vs performance
    - triggered by FFTW performance comparison in draft presentation
    - https://guix-hpc.bordeaux.inria.fr/blog/2018/01/pre-built-binaries-vs-performance/

Other software build tools

- barely used in HPC context
- lack of support for multi-user environments
- fewer supported scientific software packages

Portage - *https://wiki.gentoo.org/wiki/Portage*

- Gentoo package management system

pkgsrc - *https://www.pkgsrc.org*

- cross-platform build system
- over 15,000 supported software packages!

Homebrew - *https://brew.sh*

- "the missing package manager for macOS"
- ported for Linux: http://linuxbrew.sh
- `homebrew-science` tap is no longer maintained :(

Containers for scientific software & HPC

Singularity - *http://singularity.lbl.gov*

- "Docker for HPC" (no root daemon)
- image-based containers
- existing Docker containers can be converted to Singularity images
- *huge* uptake in last 1.5 years in HPC community
- HPCwire articles: http://tiny.cc/singularity_llc, http://tiny.cc/singularity_sc17

- strong focus on "mobility of compute"
- performance is often sacrificed for portability :(

udocker - *https://github.com/indigo-dc/udocker*

- tool to run Docker containers in user space (no root required)
- leverages other tools like Singularity, PRoot, runC
- recent HPCwire article: http://tiny.cc/hpcwire_udocker