CompBioMed Containerisation Meeting

29 March 2019 - Amsterdam

Do containers *really* relieve the burden of installing scientific software?

Kenneth Hoste (HPC-UGent, 

)

(remote talk)

email: kenneth.hoste@ugent.be - GitHub: @boegel - Twitter: @kehoste

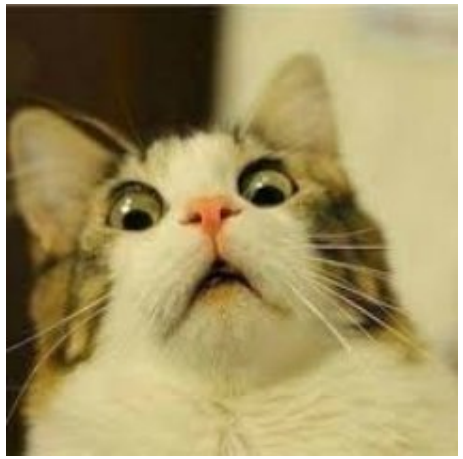
whoami

kenneth.hoste@ugent.be
@boegel *(GitHub, IRC, Slack)*
@kehoste *(Twitter)*

- Masters & PhD in Computer Science from Ghent University (Belgium)
- joined HPC-UGent team in October 2010
- main tasks: user support & training, *software installations*
- slowly also became  **lead developer & release manager**
- likes family, beer, loud music, FOSS, helping people, dad jokes, stickers, ...
- doesn't like CMake, SCons, Bazel, setuptools, software dependencies, ...

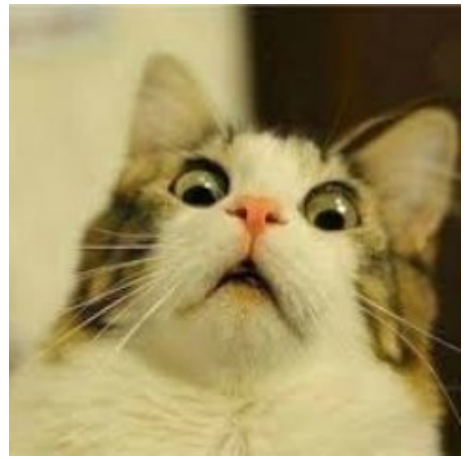
Spoilers...

Do containers *really* relieve the burden of installing scientific software?



No, not really.

(not yet?)



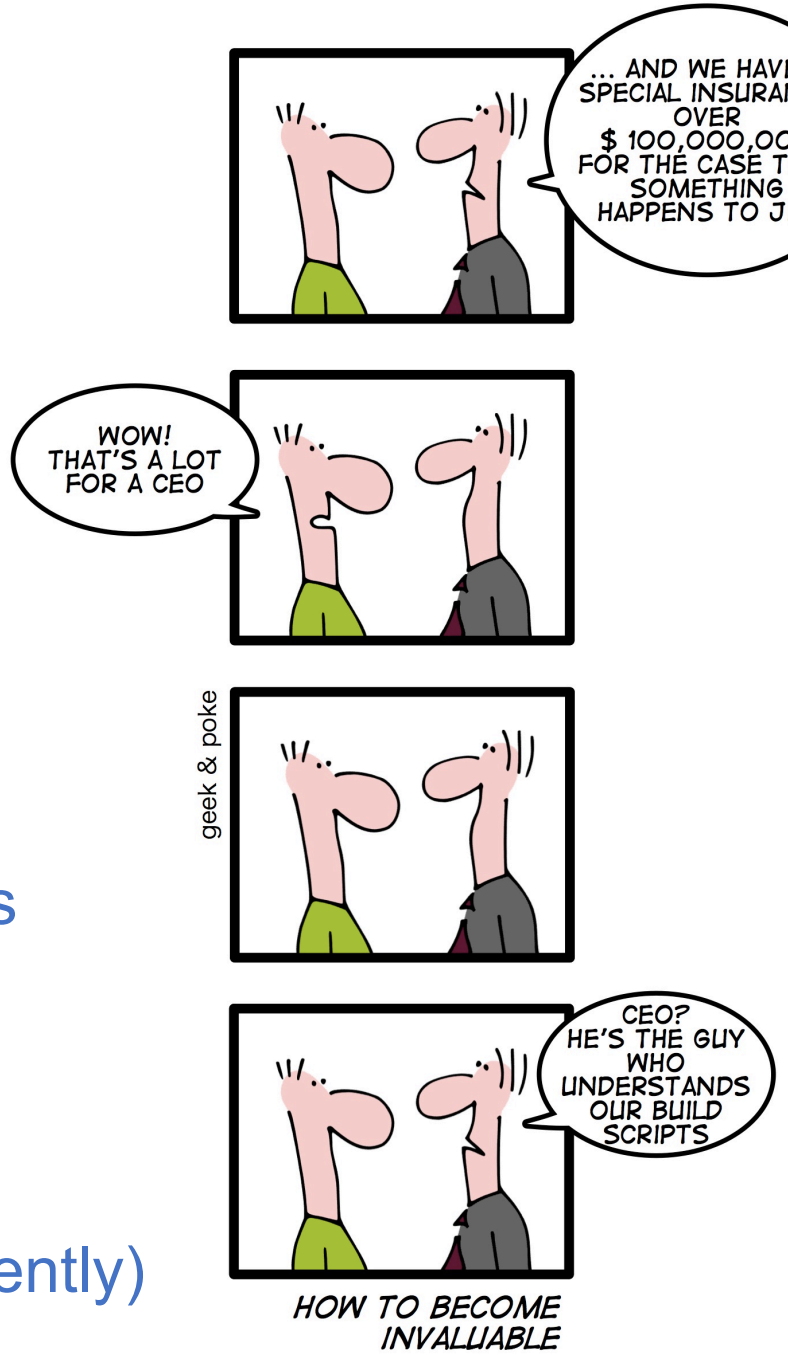
Setting the stage

- The ubiquitous problem of **getting scientific software installed**
- A short look back at the "**container revolution**" in the HPC community
  - What happened, where are we now?
  - *Why* did it happen?
- Some **concerns** on using containers for scientific software on HPC systems
- A couple of **ideas & possible solutions to improve current practice**
- Q&A

Getting scientific software installed

Installation of scientific software is a tremendous problem for HPC sites all around the world.

- ideally built from source (performance is key!)
- tedious, time-consuming, frustrating,
  sometimes simply not worth the (manual) effort...
- huge burden on researchers & HPC support teams
  - over 25% of support tickets at HPC-UGent,
    but consumes *way* more than 1/4th of support time...
- very little collaboration among HPC sites (until recently)



http://geekandpoke.typepad.com/geekandpoke/2010/05/how-to-become-invaluable.html

Common issues with scientific software

Researchers focus on the *science* behind the software they implement,
and care little about software engineering, tools, build procedure, portability, ...

Scientists are (typically) not software developers or system administrators
(nor should they be!)

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

This results in:

- use of non-standard build systems (or broken ones)
- "creative" software versioning (or no versions at all)
- dependency hell on steroids
- interactive installation scripts
- hardcoded parameters (compilers, libraries, paths, ...)
- poor/outdated/missing/incorrect documentation

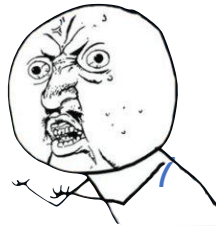
Prime example: TensorFlow

popular open-source software for Deep Learning (https://www.tensorflow.org)

- auto-installs most dependencies, but not all...
- '`configure`' script is a custom interactive script
  - silent configuration possible, but only if you *know* which `$TF_*` environment variables to set!
- uses ***Bazel*** (http://bazel.io) as build tool (there ~~is~~ was a contributed CMake alternative...)
  - *resets environment*, may result in unsetting important env. variables (e.g., `$PYTHONPATH`)
  - quite different from other build tools; e.g. to build TensorFlow:

    ```
    bazel build --config=opt //tensorflow/tools/pip_package:build_pip_package
    ```

    No, that's not a typo...
  - `--config=opt`, `-c opt` and `-copt=...` <= these options all mean different things... 0_o
- installation via '`pip install`' of locally built Python wheel file (`.whl`)



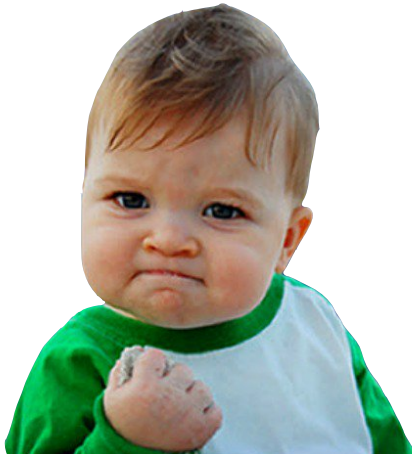
How to make package managers cry

FOSDEM 2018

https://archive.fosdem.org/2018/schedule/event/how_to_make_package_managers_cry

https://www.youtube.com/watch?v=NSemIYagjIU

- me **venting ~7 years of frustration** with getting scientific software installed
- TensorFlow & Bazel as main motivators
- **sarcastic tone for dramatic effect** (it worked!)
- lots of feedback (and ideas for an extended version of the talk), **clearly hit a nerve**...
- others discussing similar points:
  - *"Software disenchantment"*, blog post by Nikita Prokopov (Sept 2018)
    http://tonsky.me/blog/disenchantment
  - *"Don't package your libraries, write packageable libraries!"*, talk at CppCon 2018
    by Robert Schumacher (Microsoft) - *https://www.youtube.com/watch?v=sBP17HQAQjk*



What about existing software installation tools?

- package managers: *yum* (RPMs), *apt* (.deb), ...
- *Homebrew* (macOS), http://brew.sh ; *Linuxbrew*, http://linuxbrew.sh
- *Portage* (Gentoo), http://wiki.gentoo.org/wiki/Project:Portage
- *pkgsrc* (NetBSD & (a lot) more), http://pkgsrc.org

None are well suited to scientific software and HPC systems in particular.

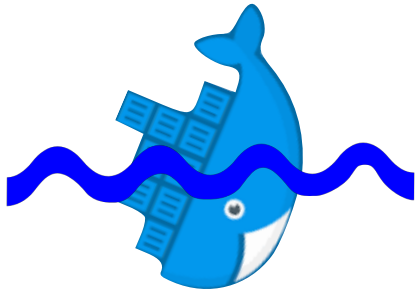
Common problems:

- usually poor support for old/multiple versions and/or to have builds side-by-side
- not flexible enough to deal with idiosyncrasies of scientific software
- little support for scientific software, non-GCC compilers, MPI, GPUs, ...
- strong focus on *generic* binaries (not optimised for *your* system architecture)

Modern exceptions: **conda, EasyBuild, Spack, Nix/Guix** (different use cases)

The (HPC) container revolution

- **containers are an old idea that was revamped (again and again)**
  - chroot (1979), FreeBSD jails (2000), Solaris Containers (2004), LXC (2008), ...
- first became (really) popular in cloud environments, hyperscale
- hype took off with Docker in 2013
- buzz around **containers for HPC** started in 2015
- first Docker tried to convince the HPC community (& failed?)
- **Singularity** (LBNL, Sylabs) **(really) started the revolution** in HPC (~2016)
  https://www.sylabs.io/singularity
- other solutions have also popped up, incl. CharlieCloud (LANL)
  https://github.com/hpc/charliecloud



Why did the (HPC) container revolution start?

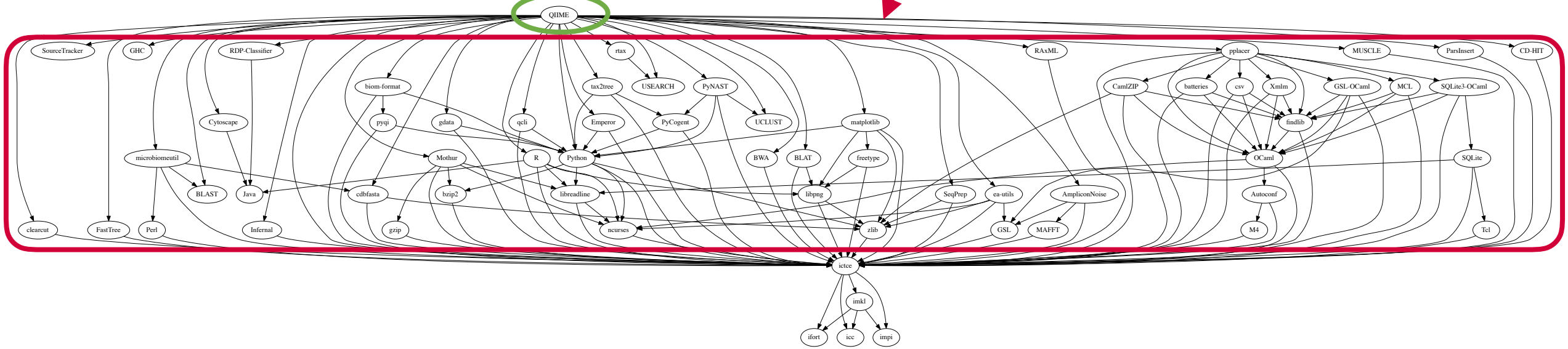
- **installing scientific software (correctly/efficiently) is becoming more complex**
  - dependency hell is getting worse *(*fast*)*
  - growing variety in target platforms (x86_64, GPUs, ARM, POWER, ...)
  - specifically targeting host system is becoming *more* important for performance
- **scientists using/developing scientific software lack expertise** in
  - (sensible) software release management
  - picking the right tool for the job (tip: not a custom Perl script named '`configure`')
  - diagnosing & fixing compilation/installation problems
- some popular programming languages & applications are not great examples...

- also, **people are lazy/impatient** (and StackOverflow doesn't always have an easy answer)

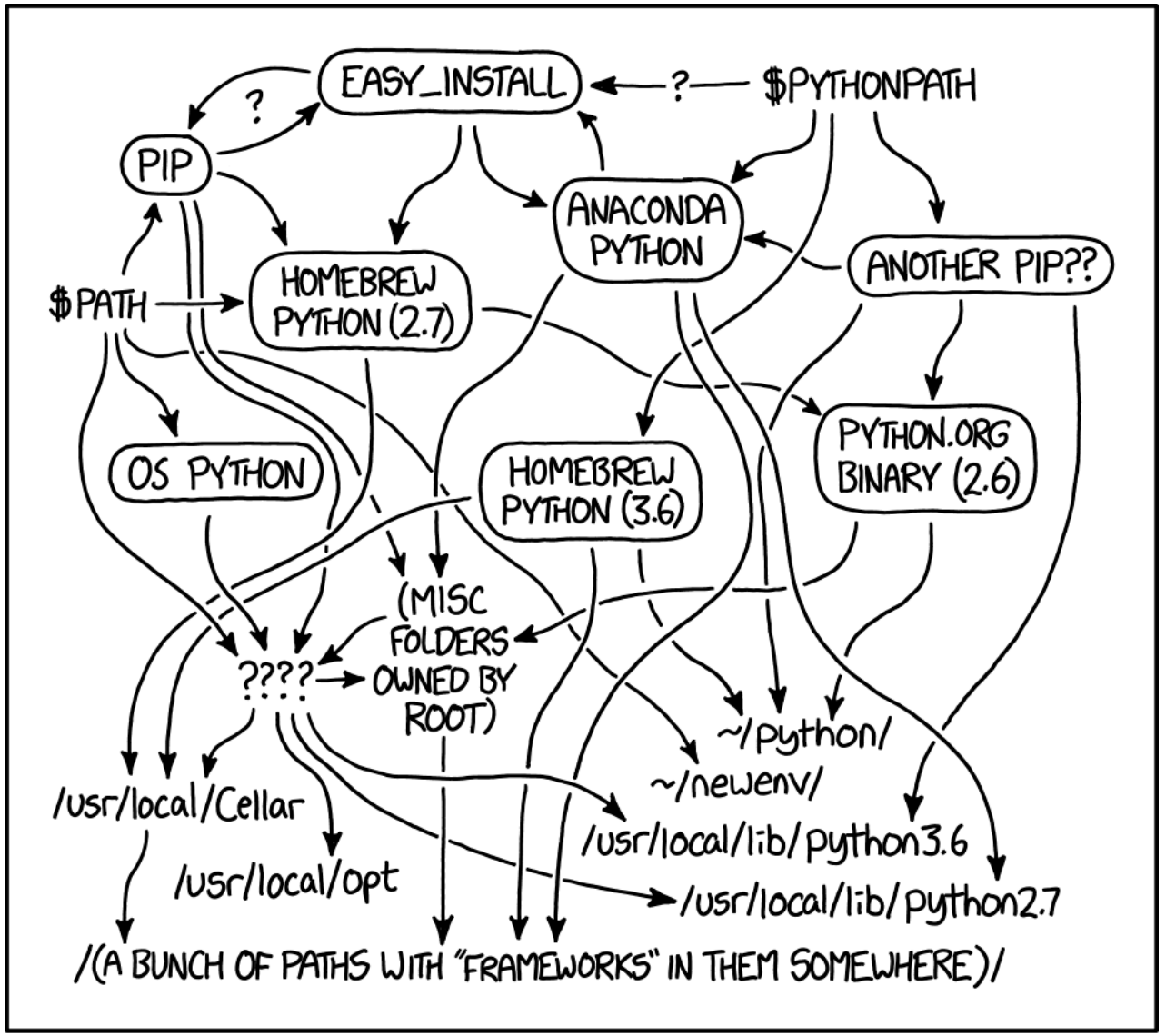
Dependency hell in scientific software

this is the part we actually care about

most of the rest is a necessary evil...



dependency graph for (old version) of QIIME (http://qiime.org)



https://xkcd.com/1987

the Python mess...

Containers for HPC: buzzword bingo!

(screenshot from Singularity website @ https://www.sylabs.io/singularity)

- simplicity
- "mobility of compute"
- "extreme mobility"
- Bring Your Own Environment (BYOE)
- "native" performance
- reproducibility
- easy integration with system resources & services
- security

So everything is rainbows & unicorns?

No, there's (still) some shit too...

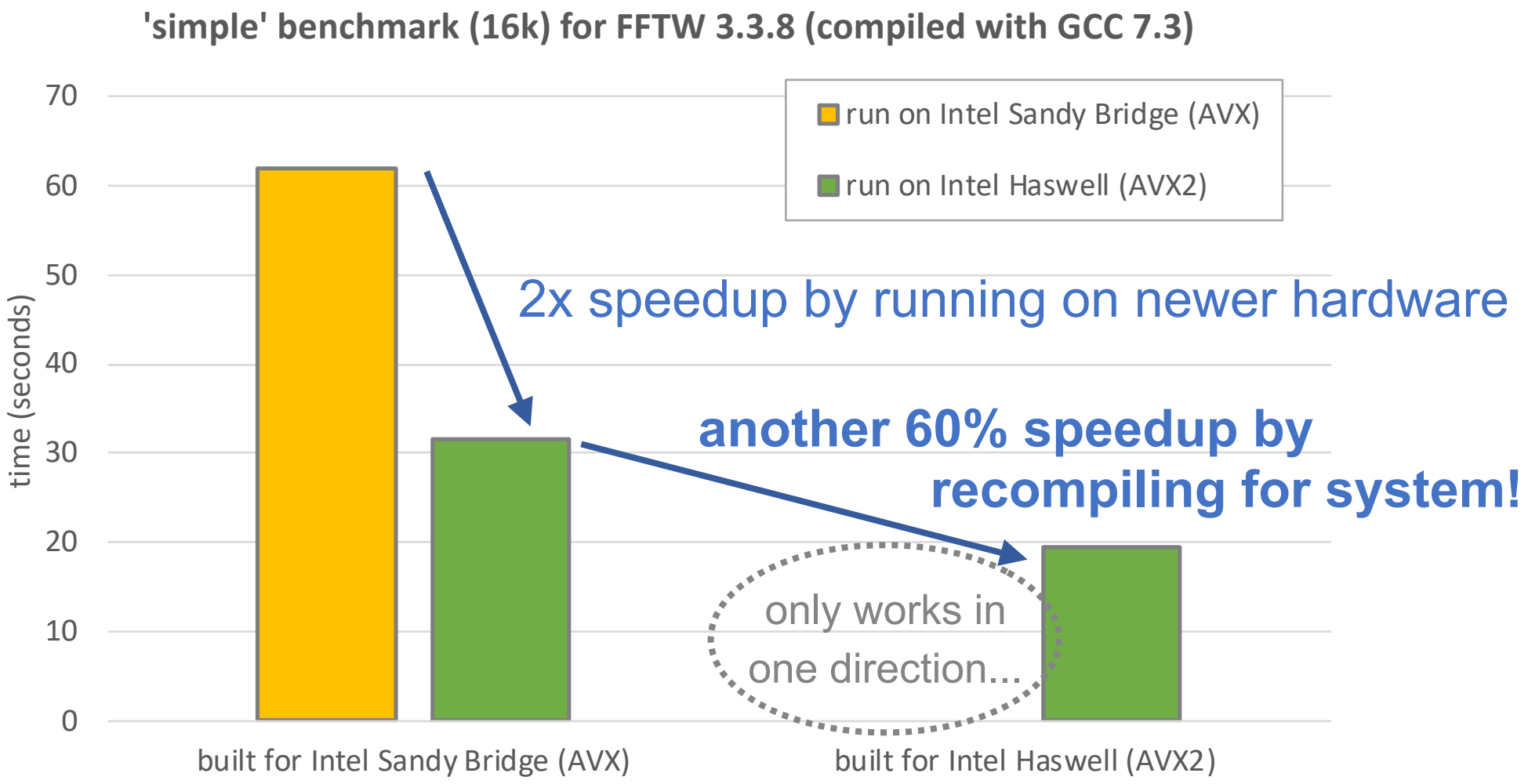
Containers are a workaround, *not a solution...* (1/2)

- **container images are basically just a fancy set of binaries...**
  - they are most likely *not* built for the processor architecture of the host
  - ... which implies you may be losing quite a bit of performance
  - containers built for x86_64 will *not* work on ARM or POWER (and vice versa)

- **"native" performance really means "very little overhead"**
  - that's great, but it's also a *requirement* for success in HPC context...
  - there's a lot more to it in order to claim *native* performance (in my opinion)

- **integration with system resources & services is not a solved problem (yet)**
  - works great for some things (home directory, resource managers like Slurm, ...)
  - problematic for others (container/system GPU/MPI libraries need to be "in sync")

Performance is traded off for "mobility of compute"



(FFTW 3.3.8 installed in Singularity container)

Containers are a workaround, *not a solution...* (2/2)

- **someone still needs to puzzle together the container image you want/need**
  - and hopefully in a way that is easy to tweak later...
  - ideally, the container is targeted to your host system
  - what if you want to update to more recent software versions?
- **we basically need a "containers on demand" service**
  - select the software (versions) & configuration you want/need
  - specify specs of your host system (processor, GPU, host libraries, ...)
  - click "Go!", wait for a while, download container image => happy computing!
- other unresolved issues/concerns: security, trust, transparency, ...
- **more work needed to make containers *truly* successful in HPC (beyond the hype)**

Some ideas to improve current practice

- **raising awareness** that containers are not all pink fluffy unicorns and rainbows...
- more **standardisation**, less chaos
- documented (general) **best practices** for building container images
- **sharing (reproducible/validated*)* container *recipes*** in an organised way
- **metadata** to capture specifics of target system for which containers were built

  \+ accompanying tooling to query that metadata
- **leveraging established tools** like 

for building containers images
- building container images as fancy sets of ***fat* binaries** (optimised for multiple targets)

https://easybuilders.github.io/easybuild - https://easybuild.readthedocs.io

- **framework for installing scientific software** *(built from source when possible)*
- strong focus on Linux & HPC systems (and hence also performance)
- **builds specifically for host architecture by default**
- implemented in Python 2, lead development by HPC-UGent
- available under GPLv2 license via PyPI, GitHub
- supports: different compilers & MPI libraries, > 1,600 different software packages
- **(experimental) support to generate (recipes for) Docker/Singularity containers**
- active & helpful worldwide community

Supported software

http://easybuild.readthedocs.io/en/latest/version-specific/Supported_software.html

- latest EasyBuild (v3.8.1) supports installing **over 1,643 different software packages**
  - including CP2K, NAMD, NWChem, OpenFOAM, TensorFlow, WRF, ...
  - also a lot of bioinformatics software is supported out of the box
  - + ~1,000 extensions: Python packages, R libraries, Perl modules, X11 libraries, ...
  - built from source when possible...
- diverse toolchain support:
  - compilers: GCC, Intel, Clang, PGI, IBM XL, Cray, CUDA
  - MPI libraries: OpenMPI, Intel MPI, MPICH, MPICH2, MVAPICH2, Cray MPI, ...
  - BLAS/LAPACK libraries: Intel MKL, OpenBLAS, ScaLAPACK, BLIS, Cray LibSci, ...

Building TensorFlow from source with one command...


```
$ eb TensorFlow-1.13.1-foss-2018b-Python-3.6.6.eb
== temporary log file in case of crash /tmp/eb-GyvPHx/easybuild-U1TkEI.log
== processing EasyBuild easyconfig TensorFlow-1.13.1-foss-2018b-Python-3.6.6.eb
== building and installing TensorFlow/1.13.1-foss-2018b-Python-3.6.6...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== patching...
== preparing...
== configuring...
== building...
== testing...
== installing...
== taking care of extensions...
== postprocessing...
== sanity checking...
== cleaning up...
== creating module...
== permissions...
== packaging...
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file /opt/easybuild/software/Tensor...
== Build succeeded for 1 out of 1
== Temporary log file(s) /tmp/eb-GyvPHx/easybuild-U1TkEI.log* have been removed.
== Temporary directory /tmp/eb-GyvPHx has been removed.
```

What  easybuild is *not*

- EasyBuild is *not* YABT (Yet Another Build Tool)

  it does *not* replace build tools like `cmake` or `make`; it wraps around them

- it is *not* a replacement for your favourite package manager (`yum`, `apt-get`, ...)

  it leverages some tools & libraries provided by the OS (glibc, OpenSSL, IB drivers, ...)

- it is *not* a magic solution to all your (software compilation/installation) problems...

  you will still run into compiler errors (unless somebody has already taken care of it)

What  easybuild is

- a **uniform interface** that wraps around software installation procedures
- a huge **time-saver**, by automating tedious/boring/repetitive tasks
- a way to provide a **consistent software stack** to your users
- an **expert system** for software installation on HPC systems
- a **platform for collaboration** with HPC sites worldwide
- a way to **empower *users* to self-manage their software stack** on HPC systems
- a tool that can be leveraged for **building *optimised* container images**

& containers: current status

- EasyBuild can **generate a container recipe/image** for a specific set of software
- the resulting recipe in turn leverages EasyBuild to install the requested software
- **fully automated building of optimised container images**
- both Docker & Singularity are supported
- current support is still *experimental* (needs more testing, has some rough edges)
- can be a starting point to tackle some of the raised concerns...
- for more information, see https://easybuild.readthedocs.io/en/latest/Containers.html

Conclusions

- **Do containers really relieve the burden of installing scientific software?**
  - No, not really (not yet).
- Containers for HPC are "booming business"
- ... but they are not all unicorns & rainbows!
- Unresolved issues:
  - (truly) native performance by better targeting to host system
  - sharing of reproducible/maintainable/validated container *recipes*
  - leveraging established tools to build container images (from source)
  - tracking of metadata for transparency w.r.t. targeted system(s)

Do containers *really* relieve the burden of installing scientific software?

Questions?

CompBioMed Containerisation Meeting

29 March 2019 - Amsterdam

Kenneth Hoste (HPC-UGent, 

)

(remote talk)

email: kenneth.hoste@ugent.be - GitHub: @boegel - Twitter: @kehoste

https://users.ugent.be/~kehoste/CompBioMed_20190329_containers.pdf