EasyBuild

Instalando Software Científico de manera fácil

Kenneth Hoste
HPC-UGent, Ghent University, Belgium
kenneth.hoste@ugent.be

`http://users.ugent.be/~kehoste/EasyBuild_intro_20161109.pdf`

EasyBuild introduction @ UGent – November 9th 2016

whoami

- PhD in Computer Science from Ghent University (Belgium)
- joined HPC-UGent team in October 2010
- main tasks: user support & training, *software installations*
- inherited maintenance of EasyBuild in 2011
- slowly also became lead developer & release manager

- e-mail: kenneth.hoste@ugent.be
- Twitter: @kehoste
- GitHub: @boegel
- IRC (Freenode): `boegel`
- Google+: kenneth.hoste@gmail.com
- Skype: `kehoste`

HPC-UGent in a nutshell

`http://www.ugent.be/hpc/en` – `http://www.vscentrum.be`

- HPC team at central IT dept. of Ghent University (Belgium)
- 7+1 team members: 1 manager, $\sim$ 1.5 user support, $\sim$ 5.5 sysadmin
- 5 Tier2 clusters + one Tier1 cluster (8.5k cores)
- $\sim 2,000$ user accounts, across all scientific domains
- tasks: hardware, system administration, user support/training, ...

- member of Flemish Supercomputer Centre (VSC)
  virtual centre, collaboration between Flemish university associations

Tasks for HPC user support teams

- resolving problems that occur when using the HPC system(s)
  - *"I lost my private key/password, and now I can't log in. Help?"*
  - *"My job crashed, and I have no idea why. What happened?"*
  - *"My stuff doesn't work anymore, and I didn't change a thing!"*
- answering questions, from simple to very technical
- **installing (scientific) software tools/libraries/applications**
- helping users improve their workflow (not necessarily by request)
- training: Linux basics, OpenMP, MPI, Python, etc.
- ~~performance analysis and optimisation of large scientific applications~~
- ~~consultancy services w.r.t. developing scientific software~~

Installing scientific software for users

Typical way in which scientific software is installed for users:

- by user request: new software, version upgrades, more variants, ...
- on a (shared NFS) filesystem available on every workernode
- specifically targetted to the HPC cluster it will be used on
  - **built from source** (if possible)
  - separate installation per cluster
  - **highly optimized** for system architecture
    - linked with heavily tuned libraries (MPI, BLAS, LAPACK, ...)
    - built with (equivalent of) `-march=native`/`-xHost`
- rebuild when updates for compilers/libraries become available
- installations remain available during lifetime of system
- accompanying *module file* is provided for easy access

Environment modules

- canonical way of giving users access to installed (scientific) software
- used on most HPC systems ($>$ 80%[1]), since mid 90's
- module file specifies changes to user environment (in Tcl/Lua subset)
- modules tool applies those changes to the current session (!)
- **easy interface for users**:
  - available software: '`module avail [name]`'
  - prepare environment: '`module load <name>/<version>`'
  - show loaded modules: '`module list`'
  - rollback changes to environment: '`module unload <name>`'
  - start afresh: '`module purge`'
- Tcl-based environment modules system is most prevalent (for now)
- **Lmod: modern modules tool, vastly improves user experience**

(1) `http://hpcugent.github.io/easybuild/files/SC15_BoF_Getting_Scientific_Software_Installed.pdf`

"Please install <software> on the HPC?"

The most common type of support request from users is to install (scientific) software; this covers over 25% of support tickets at HPC-UGent.

Installing (lots of) *scientific* software is:

- error-prone, trial-and-error
- tedious, hard to get right
- repetitive & boring (well...)
- time-consuming (hours, days, even weeks)
- frustrating (e.g., dependency hell)
- sometimes simply not worth the effort...

Common issues with scientific software

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

Scientists are not software developers or sysadmins (nor should they be).

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

This results in:

- 'incorrect' use of build tools
- use of non-standard build tools (or broken ones)
- incomplete build procedure, e.g., no configure or install step
- interactive installation scripts
- hardcoded parameters (compilers, libraries, paths, ...)
- poor/outdated/missing/incorrect documentation
- dependency (version) hell

Prime example I: WRF

Weather Research and Forecasting Model (`http://www.wrf-model.org`)
(one of the top 5 applications on Blue Waters)

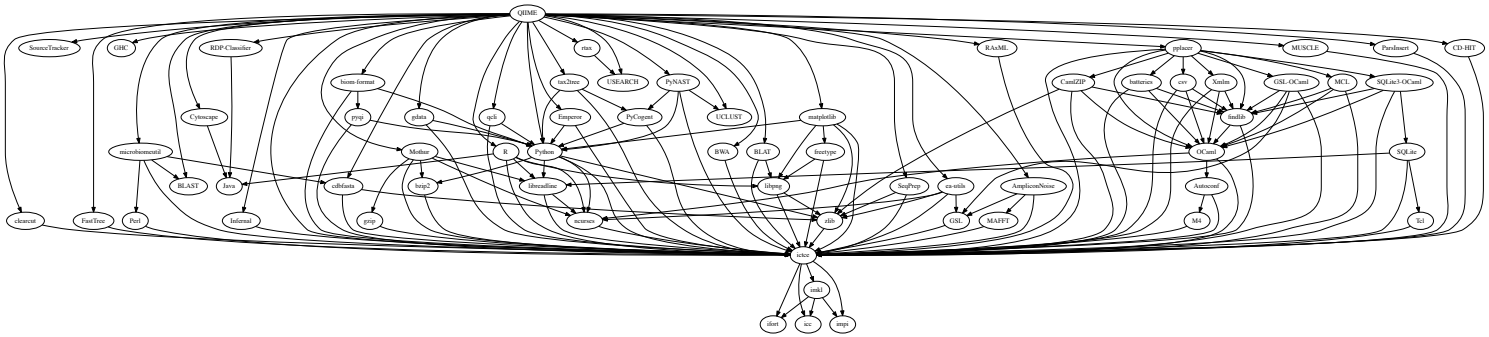
- dozen dependencies: netCDF (C, Fortran), HDF5, tcsh, JasPer, ...
- known issues in last release are (only) documented on website
  no patch file provided, infrequent bugfix releases
- interactive '`configure`' script :(
- resulting `configure.wrf` needs work:
  fix hardcoded settings (compilers, libraries, ...), tweaking of options
- custom '`compile`' script (wraps around '`make`')
  building in parallel is broken without fixing the `Makefile`
- no actual installation step

Wouldn't it be nice to build & install WRF with a single command?

`http://easybuild.readthedocs.org/en/latest/Typical_workflow_example_with_WRF.html`

Prime example II: QIIME

QIIME: Quantitative Insights Into Microbial Ecology (`http://qiime.org/`)



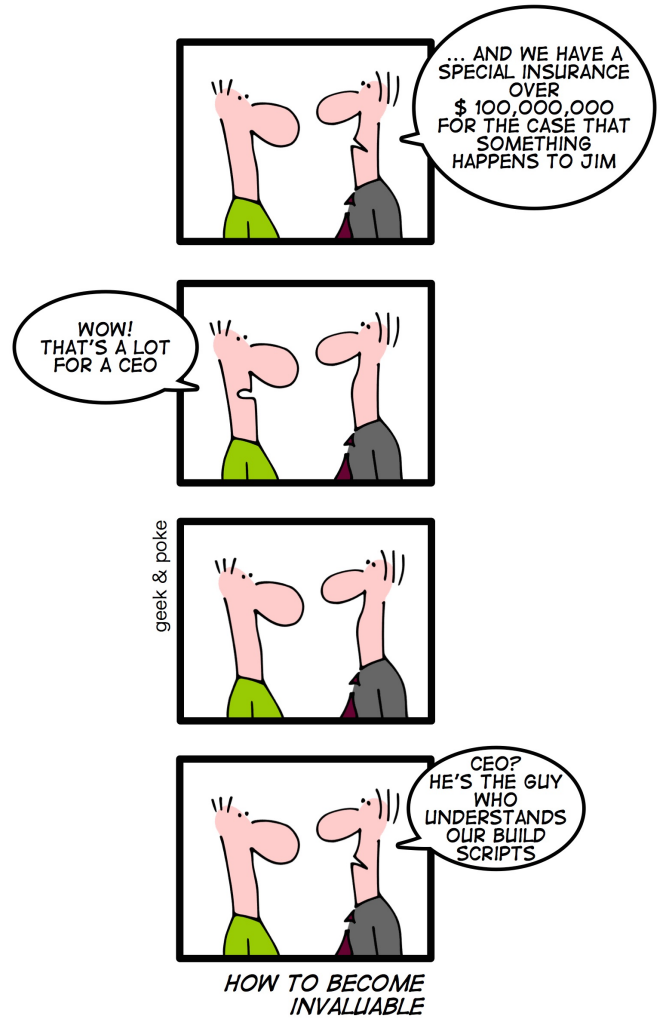
- scientific research domain: bioinformatics ...
- 59 dependencies in total (*without* compiler toolchain), some optional
  - depends on Haskell (GHC), Java, Python, R, Perl, OCaml, ...
  - several deps use a non-standard build procedure (in various degrees)
- *very* picky about dependency versions (e.g., *must* be Python v2.7.*3*)
- took us several weeks to get it installed (using Intel compilers!)...
- **...now we can (re)build/install it all with a single command!**

(disclaimer: support for QIIME not included yet in latest EasyBuild release)

Houston, we have a problem

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

- huge burden on HPC user support teams
- hard to provide a consistent software stack
- researchers lose lots of time (waiting)
- sites typically resort to in-house scripting
- very little collaboration among HPC sites :(



What about existing tools?

Existing tools are *not* well suited to scientific software and HPC systems.

- package managers: **yum** (RPMs), **apt-get** (.deb), ...
- **Homebrew** (Mac OS X), `http://brew.sh/`
- **Linuxbrew**, `http://brew.sh/linuxbrew/`
- **Portage** (Gentoo), `http://wiki.gentoo.org/wiki/Project:Portage`
- **pkgsrc** (NetBSD & (a lot) more), `http://pkgsrc.org/`
- **Nix**, `http://nixos.org/nix`
- **GNU Guix**, `https://www.gnu.org/s/guix`

Common problems:

- usually poor support for multiple versions/builds side-by-side
- not flexible enough to deal with idiosyncrasies of scientific software
- little support for scientific software, other compilers (not GCC), MPI

EasyBuild: building software with ease

http://hpcugent.github.io/easybuild/

- framework for installing (scientific) software on HPC systems
- collection of Python packages and modules
- in-house since 2009, open-source (GPLv2) since 2012
- now: thriving community; actively contributing, driving development
- new release every 6–8 weeks (latest: EasyBuild v2.9.0, Sept 23rd 2016)
- supports over 1,000 different software packages

  including CP2K, GAMESS-US, GROMACS, NAMD, NWChem, OpenFOAM, PETSc, QuantumESPRESSO, Yade, WRF, WPS, ...
- well documented: http://easybuild.readthedocs.io

EasyBuild: feature highlights

- fully **autonomously** building and installing (scientific) software
  - automatic dependency resolution
  - automatic generation of module files (Tcl or Lua syntax)
- thorough **logging** of executed build/install procedure
- **archiving** of build specifications ('*easyconfig files*')
- highly **configurable**, via config files/environment/command line
- **dynamically extendable** with additional *easyblocks*, *toolchains*, ...
- support for **custom module naming schemes** (incl. hierarchical)
- **transparency** via support for 'dry run' installations
- **comprehensively tested**: lots of unit tests, regression testing, ...
- actively developed, **collaboration** between various HPC sites
- worldwide **community**

What EasyBuild is (not)

EasyBuild is *not*:

- YABT (Yet Another Build Tool); it does *not* replace `make`, `CMake`
- a replacement for your favorite package manager
- a magic solution to all your (installation) problems

EasyBuild can be (and maybe already *is*) a:

- proper way of installing scientific software
- uniform interface that wraps around software build procedures
- **huge time-saver**, by automating tedious/boring/repetitive tasks
- way to provide a *consistent* software stack to your users
- expert system for software installation on HPC systems
- platform for collaboration with HPC sites world-wide
- tool to empower users to manage their *own* software stack

EasyBuild terminology

- EasyBuild *framework*
  - core of EasyBuild: Python modules & packages
  - provides supporting functionality for building and installing software
- *easyblock*
  - a Python module, 'plugin' for the EasyBuild framework
  - implements a (generic) software build/install procedure
- *easyconfig* file (`*.eb`)
  - build specification: software name/version, compiler toolchain, etc.
- compiler *toolchain*
  - compilers with accompanying libraries (MPI, BLAS/LAPACK, ...)

Putting it all together

The EasyBuild *framework* leverages *easyblocks* to automatically build and install (scientific) software using a particular *compiler toolchain*, as specified by one or more *easyconfig files*.

EasyBuild: system requirements

- Linux x86_64 HPC systems is main target platform (for now
  - Red Hat-based systems (Scientific Linux, CentOS, RHEL, . . .)
  - also other Linux distros: Debian, Ubuntu, OpenSUSE, SLES, . . .
  - kind of works on OS X, but not really a target platform
  - *no* Windows support (and none planned)
  - stable support for Cray systems since EasyBuild v2.7.0
  - support for Linux@POWER systems is being looked into (by TAMU)
- Python v2.6.x or more recent v2.x (not Python 3 compatible (yet))
- a modules tool:
  - latest release of Tcl/C environment modules (version 3.2.10);
  - or one of the Tcl-only versions of environment modules;
  - or a recent version of *Lmod* (5.6.3 or more recent) **(recommended!)**
- (a system C/C++ compiler, to get started)

'Quick' demo for the impatient

```
eb HPL-2.1-foss-2016a.eb --robot
```

- **downloads** all required sources (best effort)
- **builds/installs** *foss* toolchain (be patient) + HPL on top of it

  foss: GCC, OpenMPI, LAPACK, OpenBLAS, FFTW, ScaLAPACK

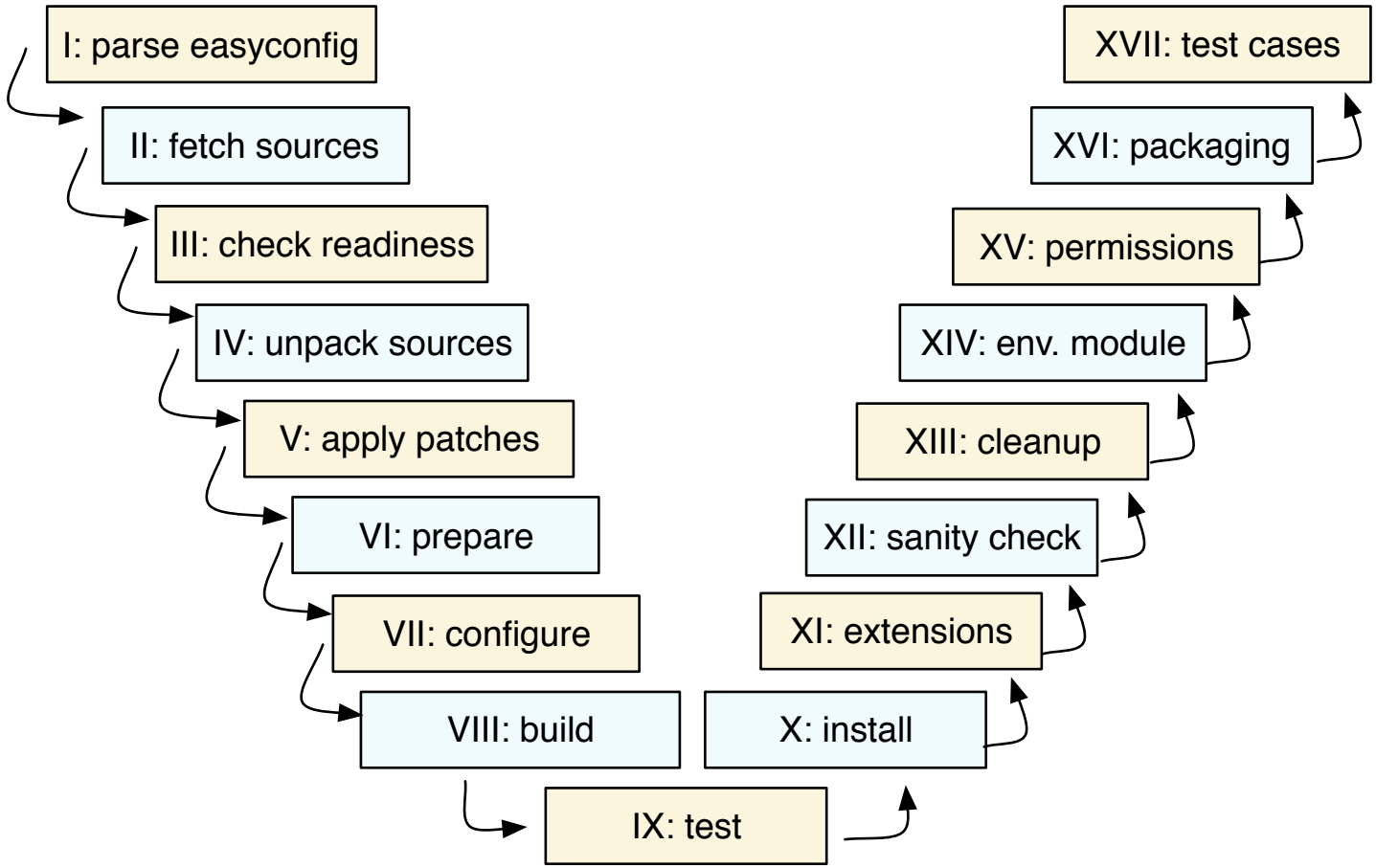
  note: requires `libibverbs` to be available
- **generates module file** for each installed software package

Example 'eb' output

```
$ eb GCC-4.9.3.eb
== temporary log file in case of crash /tmp/eb-GyvPHx/easybuild-U1TkEI.log
== processing EasyBuild easyconfig GCC-4.9.3.eb
== building and installing GCC/4.9.3...
== 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/GCC/4...
== 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.
```

Step-wise install procedure

build and install procedure as implemented by EasyBuild



most of these steps can be customised if required,
via *easyconfig parameters* or a *custom easyblock*

EasyBuild: statistics

EasyBuild v2.9.0 *(Sept'16)*

- $\sim 25,000$ LoC in framework (17 Python packages, 166 Python modules)

  $+ \sim 5,000$ LoC in `vsc-base` (option parsing/logging)
  $+ \sim 12,500$ LoC more in unit tests
  $\Longrightarrow \sim 42,500$ LoC in total

- 209 easyblocks in total ($\sim 18,000$ Loc)

  180 software-specific easyblocks
  29 generic easyblocks

- over 1,136 different software packages supported (incl. toolchains & bundles)

  bio: 286, tools: 147, vis: 117, devel: 98, lib: 104, math: 69,
  data: 61, chem: 46, toolchain: 41, lang: 39, numlib: 28,
  perf: 25, system: 21, cae: 16, compiler: 14, phys: 14, mpi: 11,
  geo: 11, debugger: 4

- 7,060 easyconfig files: different versions/variants, toolchains, ...

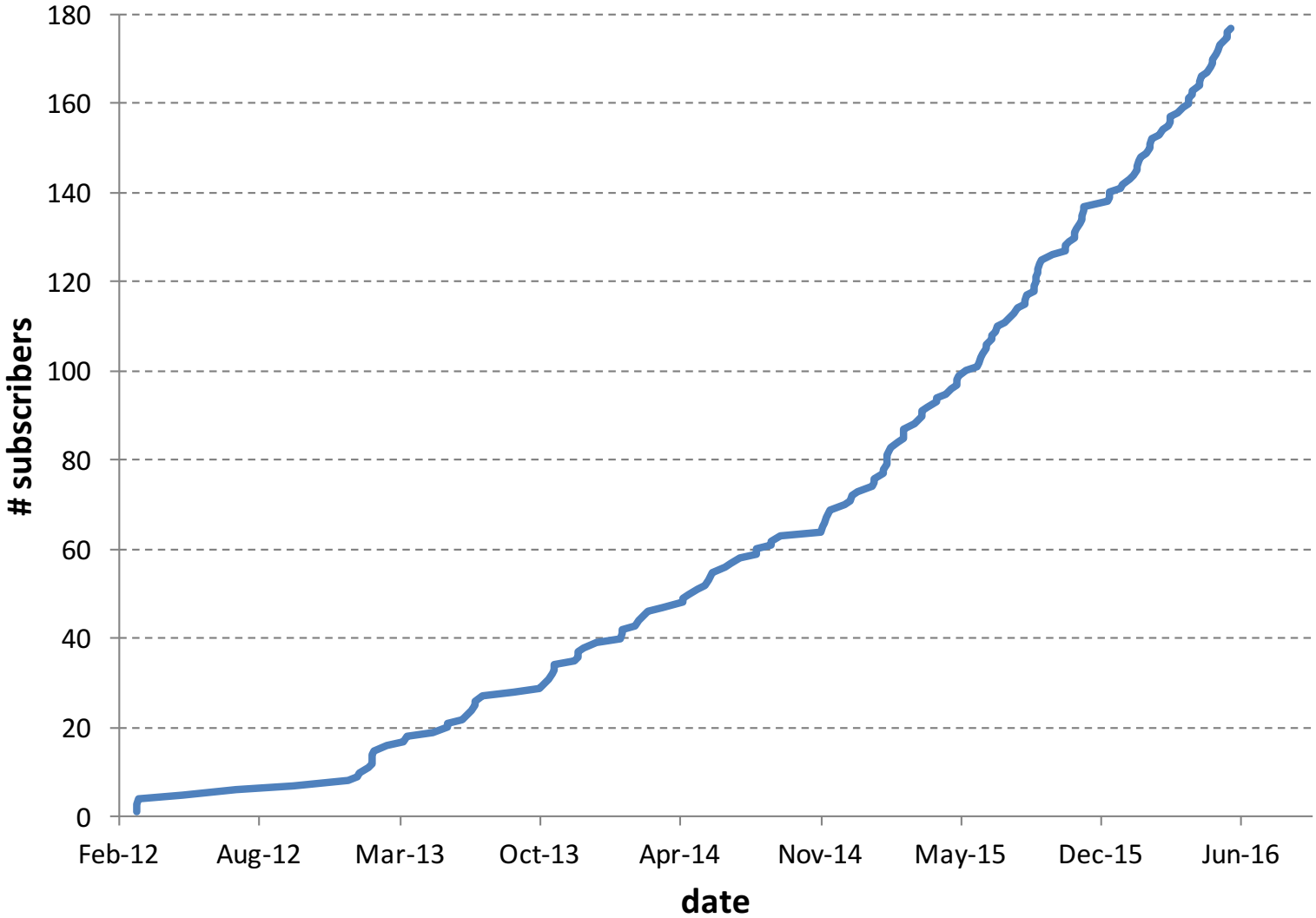
EasyBuild community

- worldwide: Europe, US, New Zealand, Australia, Cuba, ...
  - incl. large sites like Jülich Supercomputer Centre, CSCS, ...
- active mailing list & IRC channel
  - `https://lists.ugent.be/wws/info/easybuild` (recommended!)
  - `#easybuild` at `chat.freenode.net`
- bi-weekly (short) conference calls via Google Hangouts
- meetings on a regular basis
  - *Getting Scientific Software Installed* sessions at ISC/SC
  - EasyBuild hackathons
  - yearly EasyBuild User Meetings
- development is highly community driven

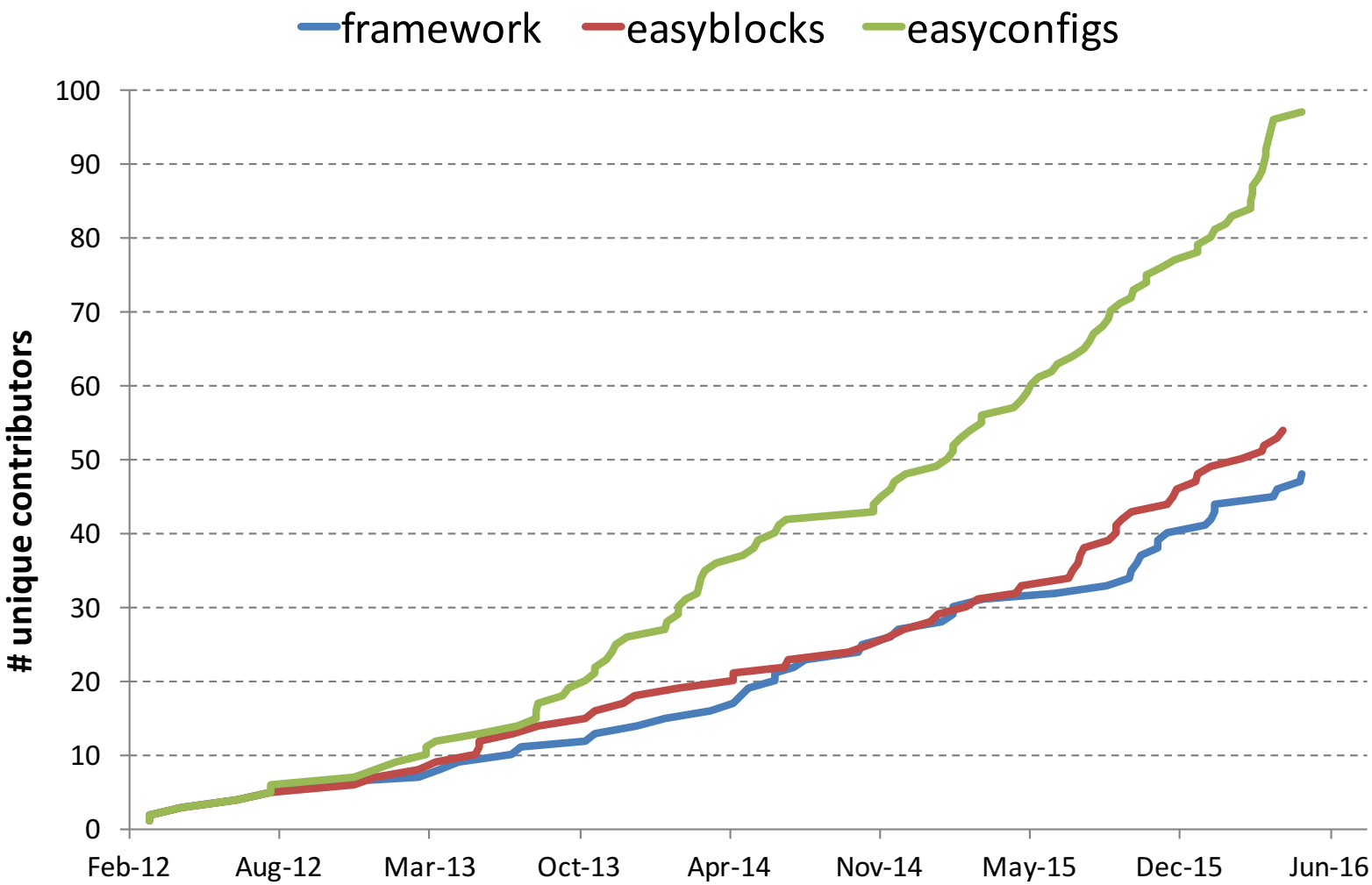
EasyBuild community by numbers

- 6 '*Getting Scientific Software Installed*' BoF sessions at ISC/SC
- 10 two/three-day EasyBuild hackathons + 1 user meeting
- ∼25 'active' souls on the `#easybuild` IRC channel
- rough estimate: about 100 HPC sites using EasyBuild
- 62 EasyBuild conference calls
- 180 subscribers to the EasyBuild mailing list
- framework: 1,153 merged PRs (51 open)
- easyblocks: 745 merged PRs (58 open)
- easyconfigs: 2,797 merged PRs (320 open)

EasyBuild mailing list



EasyBuild contributors



Recent projects similar to EasyBuild

- **Spack** (LLNL) - `http://scalability-llnl.github.io/spack/`
- **Maali** (Pawsey) - `https://github.com/chrisbpawsey/maali/`
- **Smithy** (NICS, ORNL) - `http://anthonydigirolamo.github.io/smithy/`

Major differences with EasyBuild:

- slightly different approach
- smaller community
- fewer supported software packages
- missing features
- less flexibility
- not so powerful (except Spack?)

All have expressed interest in cross-community collaboration.

Installing EasyBuild

http://easybuild.readthedocs.org/en/latest/Installation.html

Install EasyBuild using the bootstrap script *(highly recommended)*:

```
$ curl -O https://raw.github.com/hpcugent/easybuild-framework/
develop/easybuild/scripts/bootstrap_eb.py
$ python bootstrap_eb.py /tmp/$USER   # replace with your prefix!
$ module use /tmp/$USER/modules/all
$ module load EasyBuild
```

Standard Python install tools (`easy_install`, `pip`, ...) should also work.

Update EasyBuild with ... EasyBuild!

```
$ eb EasyBuild-2.9.0.eb
```

or (requires EasyBuild 2.9.0):

```
$ eb --install-latest-eb-release
```

Configuring EasyBuild

http://easybuild.readthedocs.org/en/latest/Configuration.html

By default, EasyBuild will (ab)use `$HOME/.local/easybuild`.

You should configure EasyBuild to your preferences, via:

- *configuration file(s):* key-value lines, text files (e.g., `prefix=/tmp`)
- *environment variables* (e.g., `$EASYBUILD_PREFIX` set to `/tmp`)
- *command line parameters* (e.g., `--prefix=/tmp`)

Consistency across these options is guaranteed (see `eb --help | tail`).

Priority among different options: cmdline, env vars, config file.
For example:

- `--prefix` overrules `$EASYBUILD_PREFIX`
- `$EASYBUILD_PREFIX` overrules `prefix` in configuration file

Use `eb --show-config` to get an overview of the current configuration.

Basic usage

http://easybuild.readthedocs.org/en/latest/Using_the_EasyBuild_command_line.html

- specify software name/version and toolchain to 'eb' command
- commonly via name(s) of easyconfig file(s):

```
eb GCC-4.9.2.eb Clang-3.6.0-GCC-4.9.2.eb
```

- or directory name(s) providing set(s) of easyconfig files:

```
eb $HOME/myeasyconfigs
```

- or via command line options:

```
eb --software-name=GCC
```

- `--robot`/`-r`: dependency resolution, `--debug`/`-d`: debug logging

```
eb WRF-3.6.1-foss-2015a-dmpar.eb -dr
```

Typical workflow

http://easybuild.readthedocs.org/en/latest/Typical_workflow_example_with_WRF.html

- searching for available easyconfigs (case-insensitive):

```
eb -S hpl
```

- pick an easyconfig file, based on software version, toolchain, etc.

- overview of required/available modules (dry run):

```
eb HPL-2.1-foss-2016a.eb -D
```

- enable debug logging & dependency resolution ('robot' mode):

```
eb HPL-2.1-foss-2016a.eb -dr
```

Creating easyconfig files

http://easybuild.readthedocs.org/en/latest/Writing_easyconfig_files.html

- install another software version, use different compiler toolchain, tweak build options, ... $\Longrightarrow$ new easyconfig file
- for simple changes, use `--try-*` command line options
  - different software version: `--try-software-version`
  - different compiler toolchain: `--try-toolchain`
  - the `--try-*` options integrate with `--robot`
- or copy existing easyconfig file, adjust as needed
  - 'non-trivial' changes: update dependencies, change build options, ...
  - for new software packages
- 28 generic easyblocks support over 80% of supported software!

Easyconfigs vs easyblocks

- thin line between 'fat' easyconfigs and (software-specific) easyblock
- easyblocks are "do once and forget"
- central solution for build peculiarities
- easyblocks allow to significantly simplify easyconfigs
- implemented in Python on top of framework API: very flexible
- reasons to consider an easyblock alongside a simple easyconfig:
  - 'critical' values for easyconfig parameters
  - configure/build/install options that are toolchain-dependent
  - custom (configure) options for included dependencies
  - hackish usage of parameters for existing (generic) easyblocks

`eb --extended-dry-run` (or `eb -x`)

```
eb WRF-3.6.1-foss-2015a-dmpar.eb -x
```

- figure out how EasyBuild is going to perform an installation
- runs in seconds, reports install procedure that *would* be performed
- not 100% accurate, but close enough to be practical
- opens up 'black box' that EasyBuild is sometimes perceived to be
- useful for:
  - more quickly debugging easyconfigs/easyblocks
  - understanding the effect of tweaking parameters/options
  - giving (new) users more confidence about using EasyBuild
- implemented by popular request from (new) community members

Dealing with errors

- EasyBuild is not perfect (and it will probably never be)
- most frequent issues:
  - failure to download source tarballs/installation files
    - required files can be provided manually in EasyBuild source path
  - build failures of old software versions on recent systems
    - reconsider upgrading to newer version(s)
    - see if issue can be side-stepped with a minor tweak (Google...)
  - missing dependency specification in EasyBuild
    - report problem (via GitHub issue tracker)
    - maybe try fixing it yourself, & contribute back?
  - (un)known bug in EasyBuild
- consult EasyBuild log file, enable `--debug` if needed

  `http://easybuild.readthedocs.org/en/latest/Logfiles.html`
- getting help: EasyBuild mailing list, IRC, bi-weekly conf calls, ...

  `http://easybuild.readthedocs.org/en/latest/#getting-help`

Contributing back

- contribute back your working easyconfigs/easyblocks!
- also: report bugs, share ideas for enhancements, patches, etc.
- share your expertise with the community, avoid duplicate work
- especially if:
  - software package is not supported yet
  - changes are required for a new version/toolchains
  - it is frequently used software package (compilers, MPI, etc.)
- requires a limited amount of knowledge of Git/GitHub
- contributions are reviewed & thoroughly tested prior to inclusion
- currently experimental: `eb --new-pr`, `eb --update-pr`
- see EasyBuild wiki for detailed walkthrough & guidelines:

`https://github.com/hpcugent/easybuild/wiki/Contributing-back`

`https://github.com/hpcugent/easybuild/wiki/Policy-for-easyconfig-pull-requests`

Advanced features

- upload test reports for GitHub pull requests: `eb --from-pr <id> -u`
- support for custom module naming schemes
  - make EasyBuild spit out module files following your site policy
  - also supports *hiearchical* module naming schemes
- plug in your own easyblocks, compiler toolchain definitions, etc.
  - `--include-easyblocks`, `--include-toolchains`, ...
- stable support for installing software on Cray systems
  - on top of Cray-provided modules (or not)
  - currently supports `PrgEnv-gnu`, `PrgEnv-intel`, `PrgEnv-cray`
  - sensible way to align software stacks across Cray systems/sites
- the EasyBuild community ...

Papers on EasyBuild (& Lmod)

Modern Scientific Software Management Using EasyBuild and Lmod
Markus Geimer (JSC), Kenneth Hoste (HPC-UGent), Robert McLay (TACC)
http://hpcugent.github.io/easybuild/files/hust14_paper.pdf

Making Scientific Software Installation Reproducible On Cray Systems Using EasyBuild
P. Forai (IMP), G. Peretti-Pezzi (CSCS), K. Hoste (HPC-UGent)
https://cug.org/proceedings/cug2016_proceedings/includes/files/pap145.pdf

Scientific Software Management in Real Life: Deployment of EasyBuild on a Large Scale System
D. Alvarez, A. O'Cais, M. Geimer (JSC), K. Hoste (HPC-UGent)
http://hust16.github.io/#program

Do you want to know more?

- EasyBuild website: `http://hpcugent.github.io/easybuild`
- EasyBuild documentation: `http://easybuild.readthedocs.io`
- stable EasyBuild releases: `http://pypi.python.org/pypi/easybuild`

  EasyBuild framework: `http://pypi.python.org/pypi/easybuild-framework`

  easyblocks: `http://pypi.python.org/pypi/easybuild-easyblocks`

  easyconfigs `http://pypi.python.org/pypi/easybuild-easyconfigs`
- source repositories on GitHub

  EasyBuild meta package + docs: `https://github.com/hpcugent/easybuild`

  EasyBuild framework: `https://github.com/hpcugent/easybuild-framework`

  easyblocks: `https://github.com/hpcugent/easybuild-easyblocks`

  easyconfigs: `https://github.com/hpcugent/easybuild-easyconfigs`
- EasyBuild mailing list: easybuild@lists.ugent.be

  `https://lists.ugent.be/wws/subscribe/easybuild`
- Twitter: @easy_build
- IRC: `#easybuild` on `chat.freenode.net`