Making Scientific Software Installation Reproducible On Cray Systems Using EasyBuild

Petar Forai (IMP), Kenneth Hoste (UGent),
Guilherme Peretti-Pezzi (CSCS), Brett Bode (NCSA)

May 11th – Cray User Group 2016 – London

The Ubiquitous Burden of Getting Scientific Software Installed

One particularly time-consuming task for HPC support teams is installing software on (Cray) systems.

Science teams demand rich and up to date software environment to be provided on the system(s).

Not simple to do because

- Packaged (distribution supplied) software is often out of date and a clear upgrade path is not provided by Cray during the system's life cycle.
- Scientific software installation is non-trivial in itself.

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- poor/outdated/missing/incorrect documentation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

Example: WRF

Weather Research and Forecasting Model (`http://www.wrf-model.org`)

- 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`

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.
- researchers lose lots of time (waiting).
- sites typically resort to in-house scripting (or worse).
- very little collaboration among HPC sites :(
- especially hard to reproduce builds at later point in time.

Also true on Cray systems, despite the extensive programming environment that Cray provides and similarity of software environment across installations.

Automating the build

Generic package managers and tools are *not* well suited to scientific software and HPC systems.

- package managers: **yum** (RPMs), **apt-get** (.deb), ...
- **pkgsrc** (NetBSD & (a lot) more), `http://pkgsrc.org/`
- **Nix**, `http://nixos.org/nix`
- **GNU Guix**, `https://www.gnu.org/s/guix`
- **Spack** (LLNL) - `http://scalability-llnl.github.io/spack/`

Nor do they offer integration with Cray supplied software stack (PE).

HPC specific build automation/package managers are required.

EasyBuild: building software with ease

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

- framework for installing (scientific) software on HPC systems
- implemented as Python packages and modules
- Started at University of Gent, Belgium, in 2009, open-source (GPLv2) since 2012
- now: thriving community; actively contributing, driving development
- new release every 6–8 weeks (latest: EasyBuild v2.7.0, Mar 20th 2016)
- supports over 850 different software packages

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

EasyBuild: feature highlights

- fully **autonomously** building and installing (not only scientific) software.
  - automatic dependency resolution.
  - full integration with *Environment Modules* (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*, etc.
- **Reproducibility** of the installation as one of the major design goals.
- **comprehensively tested**: lots of unit tests, regression testing, ...
- actively developed, **collaboration** between various HPC sites (worldwide **community**)
- Extensive **transparency** through verbose dry-runs and preview of all steps involved in the installation.

EasyBuild: statistics

EasyBuild v2.7.0 *(Mar'16)*

- $\sim 25,000$ LoC in framework (17 Python packages, 160 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

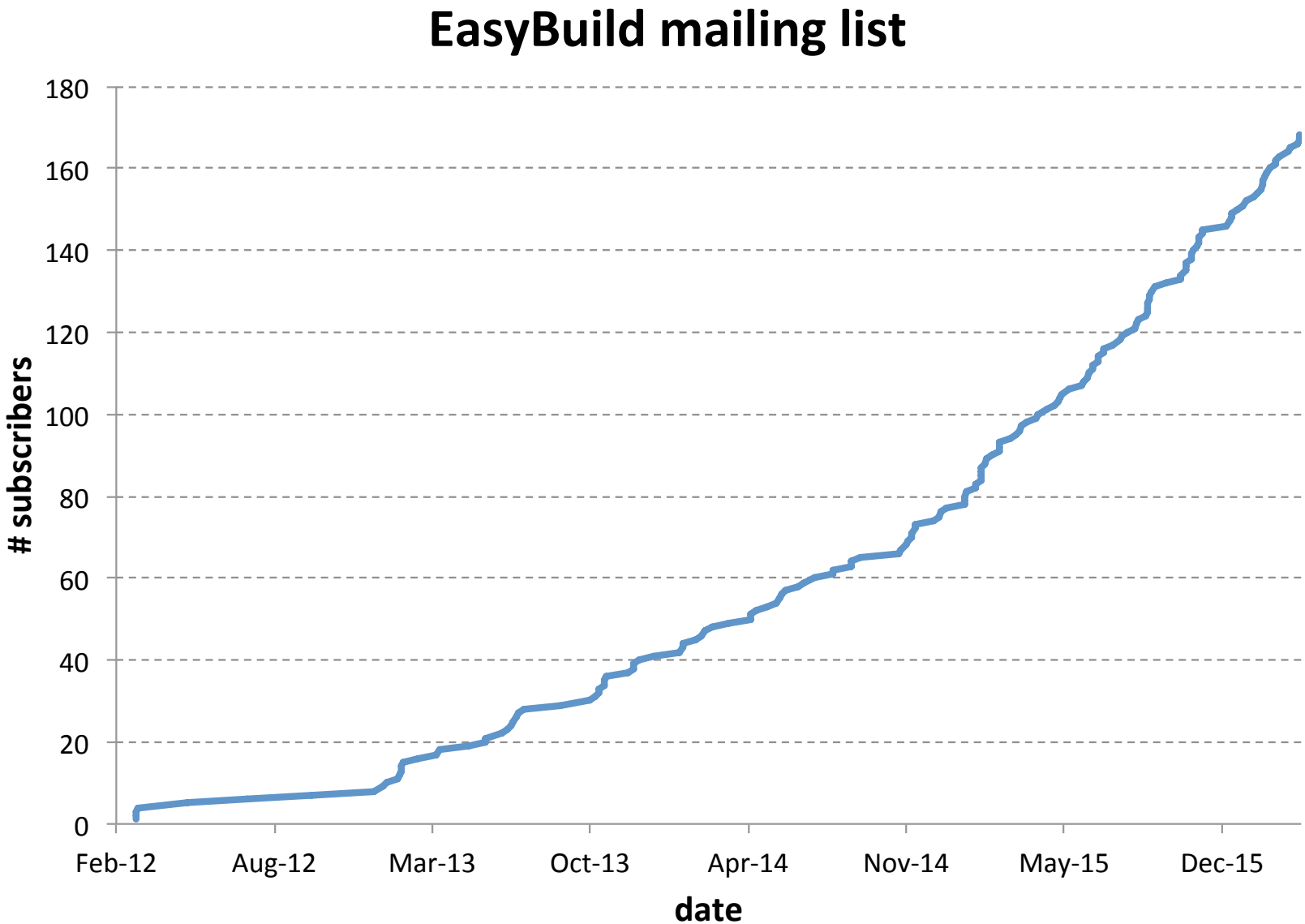
- 194 easyblocks in total ($\sim 18,000$ Loc)

  165 software-specific easyblocks
  29 generic easyblocks

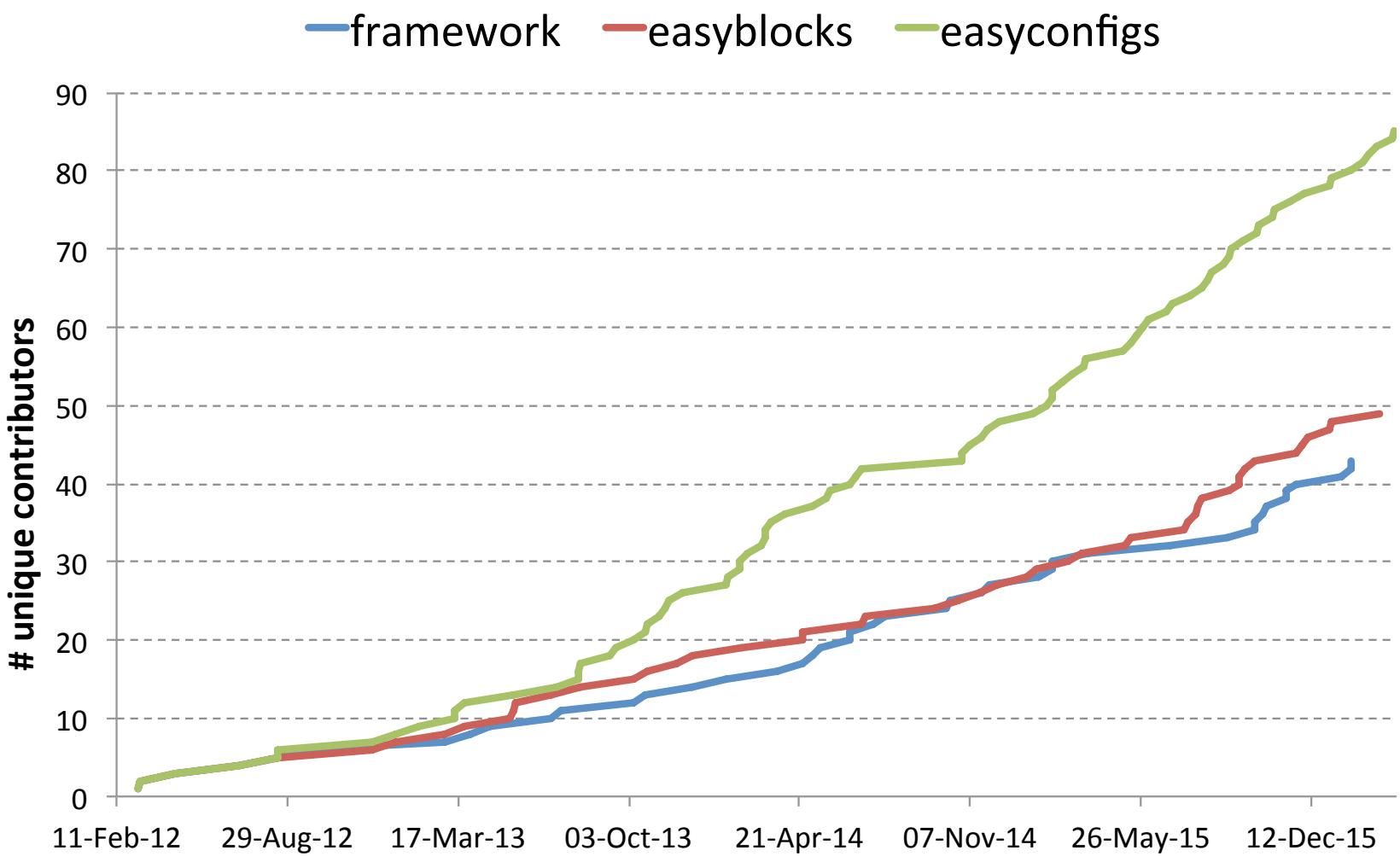
- 909 different software packages supported
  (incl. toolchains & bundles)

  bio: 203, tools: 123, vis: 99, devel: 78, lib: 77, math: 54,
  data: 53, toolchain: 38, chem: 38, lang: 32, numlib: 25,
  perf: 22, system: 21, cae: 16, compiler: 14, mpi: 11, phys: 6

- 5,580 easyconfig files: different versions/variants, toolchains, ...


EasyBuild mailing list
subscribers
0
20
40
60
80
100
120
140
160
180
Feb-12
Aug-12
Mar-13
Oct-13
Apr-14
Nov-14
May-15
Dec-15
date

EasyBuild contributors



EasyBuild terminology

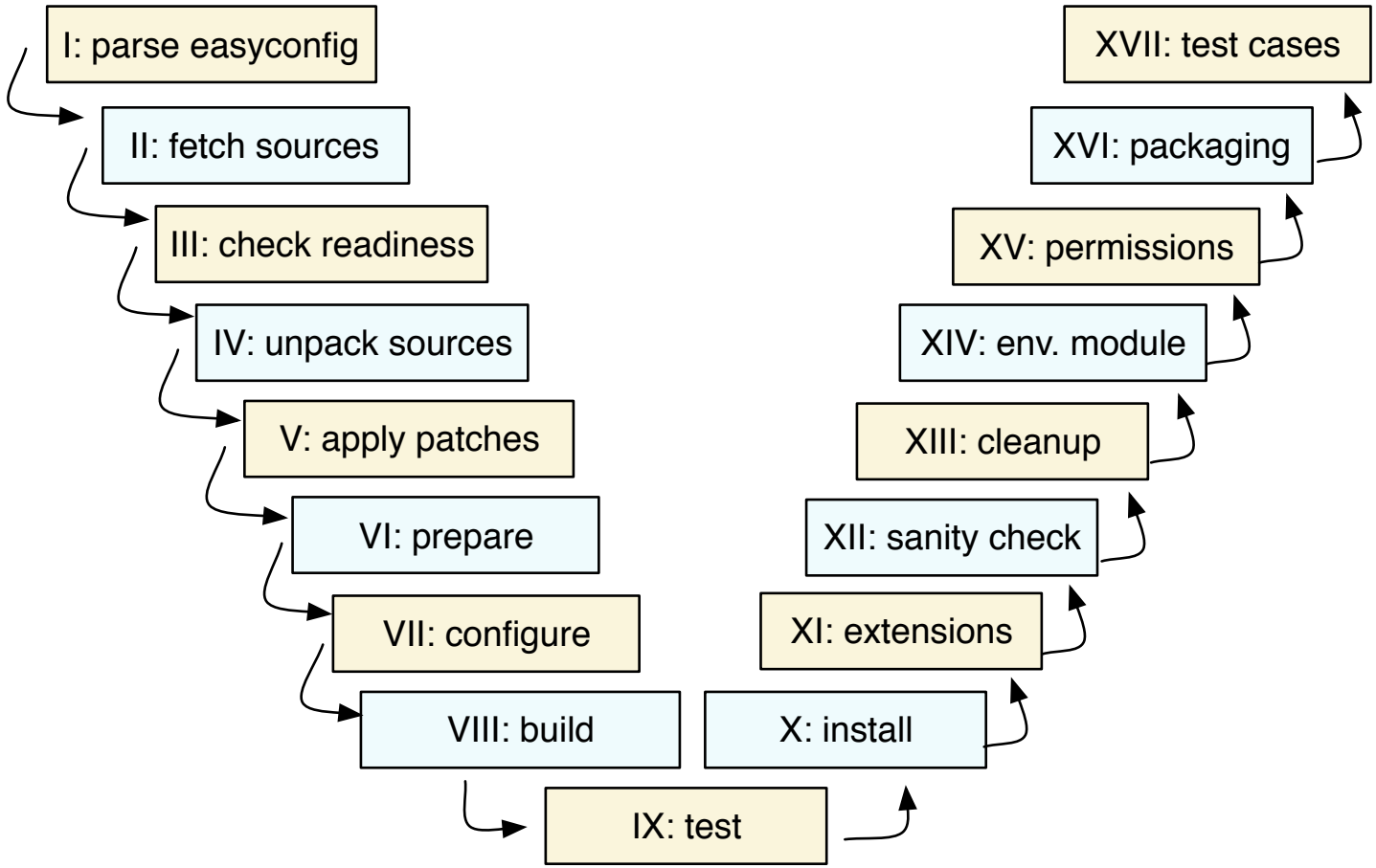
- EasyBuild *framework*
  - core of EasyBuild: Python modules & packages
  - provides supporting functionality for building and installing software
- *easyblock*
  - a Python module, 'plugin' for the EasyBuild framework
  - implements a (generic) software build/install procedure
- *easyconfig* file (`*.eb`)
  - 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*.

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 – Before Cray support

Originally designed to target generic Linux x86_64 HPC systems and not suited to an environment where compilers and libraries would be supplied by the system's vendor.

- impossible to use Cray provided programming environment easily.
- not possible to produce fully optimised installations.
- additional software and libraries supplied by Cray could not be used as build dependencies out of the box.

EasyBuild Enhancements for Cray Systems

In order to integrate EasyBuild with the Cray software stack the following enhancements were implemented and will be discussed in the following slides.

- support for external module files.
- definition of Cray-specific toolchains.
- custom easyblock for Cray toolchains.
- various smaller enhancements specific to the Cray environment.

The development of these features started early 2015, the experimental Cray support was included in EasyBuild version 2.1.0 (Apr'15).
Maturity of the EasyBuild framework allowed all those to be implemented in around 500 lines of Python.

Support for external module files

EasyBuild relies on the modules in a fundamental way as they contain information about the installed software they correspond to.

- dependency resolution mechanism uses the environment modules to gather information about software already available.
- EasyBuild can leverage modules that were not generated by EasyBuild for example as part of the Cray PE.
- this includes support that was added to supply metadata for external modules, so that EasyBuild can be made aware of
  - the software name(s), version(s)
  - installation prefix
- since EasyBuild version 2.7.0, a file containing metadata for selected modules provided by the Cray PE is included as part of EasyBuild.

Definition of Cray-specific Toolchains

Three different Cray-specific toolchains have been implemented.
Each of them corresponds directly to one of the `PrgEnv` modules:

- `CrayCCE` for `PrgEnv-cray`
- `CrayGNU` for `PrgEnv-gnu`
- `CrayIntel` for `PrgEnv-intel`

- compiler component of the toolchains EasyBuild leverages the compiler wrappers provided by the Cray PE and EasyBuild exposes
  - `$CC`, `$CXX`, `$CFLAGS`, `$CXXFLAGS`, `$F77`...
- toolchain consists of the respective compiler suite (Cray Compilation Environment, GCC, Intel) together with the Cray MPT library (provided by the `cray-mpich` module) and Cray LibSci libraries (`cray-libsci`)
- LibSci and MPT have been added through the use of external modules.
- defaults to dynamic linking through the use of `$CRAYPE_LINK_TYPE`

Custom easyblock for Cray toolchains

- support for the toolchains is implemented as an easyblock. This easyblock defines the **version pinned** components that make up the toolchain.
- new toolchain version combinations can then be placed in easyconfig file to ease creation of new toolchains.
- Easyblock implements logic to render the module files for the EasyBuild Cray toolchains.
- those are specially crafted to ensure the switching between toolchain variants works reliable even on Cray systems with complex module file logic.
- Avoids the need to run `module purge` and makes sure the desired `PrgEnv` module is loaded when EasyBuild runs independent of which modules were loaded previously.

Version Pinning of Toolchain Components

- toolchain version is based on PE release document, ie for PE release of November 2015 the corresponding GCC based toolchain is called **CrayGNU/2015.11**
- No need to discriminate between different types of Cray systems (e.g. XC vs XE/XK) wrt to toolchain versioning and components.
- Sites are expected to agree upon exact versions of subcomponents if multiple versions are provided with PE update release, otherwise latest is picked.
- components that are version pinned include all modules that influence the ABI of the generated binaries. This includes
  - the compiler
  - the numerical libraries (LibSci)
  - MPI (Cray MPT)

  but leaves the loaded modules for `PrgEnv` and `craype` at the system's default versions.

Various Smaller Enhancements for the Cray Environment

- control of CPU targeting is implemented through the load of the user desired `craype-target` module and allows for cross compilation.
- EasyBuild never runs '`module purge`' in an environment defined using Cray PE modules since this can not be done reliably.
- after EasyBuild finishes a build the initial environment is fully restored.

Testing and Evaluation – Systems

The support in EasyBuild version 2.7.0 for integrating with the Cray PE has been tested and evaluated on various Cray systems, using multiple established scientific applications.

- *Piz Daint & co – CSCS, Switzerland*
- *Sisu – CSC, Finland* Cray XC40 series system hosted at the IT Center for Science (CSC).
- *Swan – Cray, US* Cray XC40 series system hosted at Cray.
- *Blue Waters – NCSA, US* Cray XE6/XK7 system hosted at National Center for Supercomputing Applications.
- *Titan – ORNL, US* Cray XK7 system operated by the Oak Ridge Leadership Computing Facility.

Testing and Evaluation – Software

- *CP2K* tested version 2.6.0 was tested, as well as the distributed (`psmp`) variant of CP2K version 3.0.
- *GROMACS* Version 4.6.7 using the `CrayGNU` and `CrayIntel` toolchains.
- *WRF* Version 3.6.1 tested with `CrayGNU` and `CrayIntel` toolchains.
- *Python* Various recent versions of Python together with popular Python packages versions of the `CrayGNU` toolchain;
- *DCA++ dependencies* tested using the `CrayGNU` toolchain.

Where applicable, Cray-provided modules are leveraged.

EasyBuild at CSCS

CSCS began evaluating EasyBuild with release of version 2.1 (Apr'15), when the first experimental support for Cray PE integration.

- scientific software installation on the diverse pool of HPC systems hosted at CSCS was done with little or no automation.
- expertise on installing complex applications was scattered across
  - various individuals
  - personal scripts
  - support tickets
- EasyBuild was evaluated with the deployment of a consistent Python environment across all systems operated by CSCS.

EasyBuild at CSCS – Success Story

- EasyBuild evaluation experience was very positive, as a fully fledged Python installation built on top of optimised Cray PE components was provided to end users in a matter of days.
- central repository for all software installations at the center was created and made accessible to all staff of the HPC user support team.
- EasyBuild has helped signicantly in providing more consistency for both users and the CSCS support team, across various HPC systems hosted at CSCS.
- CSCS support team only provides full support for a select set of applications, EasyBuild has enabled to provide additional limited support to users for installing other software as well since easyconfigs are shared with scientific users.

EasyBuild at CSCS – Systems

EasyBuild is deployed on a center-wide central shared file system (GPFS) on most of the production and test systems at CSCS, including

- Piz Daint: Cray XC30 with 5,272 compute nodes (Intel Sandy Bridge + NVIDIA Tesla K20X)
- Piz Dora: Cray XC40 with 1,256 compute nodes (Intel Haswell); extension to Piz Daint
- Santis & Brisi: Test and Development Systems (TDS) for Piz Daint and Piz Dora
- Pilatus: Linux HPC cluster with 44 compute nodes (Intel Sandy Bridge-EP)
- Mönch: Linux HPC cluster with 440 compute nodes (Intel Ivy Bridge)
- Kesch & Escha: two Cray CS-Storm cabinets operated for MeteoSwiss (Intel Haswell + NVIDIA Tesla K80);

EasyBuild at CSCS – Use Case: CS-Storm

MeteoSwiss, the Swiss national weather forecasting service, hosts their dedicated production systems at Cray CS-Storm at CSCS.

- only `PrgEnv-cray` based part of the Cray PE release is supported on those machines. Therefore, CSCS opted to install their own software stack from scratch.
- soon it became clear that the system s provided GCC-based compiler stack was unable to assemble optimised (AVX2) instructions for system's Intel Haswell processor.
- could be solved by including a more recent version of GNU binutils that has an Haswell aware assembler.
- whole software stack required for the deployment was included in EasyBuild and the machine's environment prepared for production use within several days without having to wait for an upstream fix.
- EasyBuild was already tested on the XC series of machines at CSCS a final decision to switch all software deployments to EasyBuild was finalized.

EasyBuild at CSCS – Use Case: Continuous Testing I

- during the life cycle of an HPC system, updates to the machine environment (software, firmware, hardware) are inevitable.
- Cray PE updates appear as frequently as every month.
- qualifying software and hardware updates on dedicated test and development systems is considered good practice.
- should happen as frequently as possible, and ideally as an continuous process.
- all steps in software deployment process should be designed to run fully automated.

- CSCS decided on using the Jenkins project.
- Automatically executed building of software delivered via EasyBuild based on commits to central source code repositories.

EasyBuild at CSCS – Use Case: Continuous Testing II

- The Jenkins dashboard allows every stakeholder to quickly understand the capacity of the entire software stack to be reinstalled on every system the center operates.

		RegressionEBBrisi	20 hr - #8	N/A	1 hr 9 min
		RegressionEBDaint	21 hr - #11	N/A	1 hr 49 min
		RegressionEBDora	21 hr - #22	N/A	1 hr 43 min
		RegressionEBEscha	15 days - #8	N/A	4 hr 21 min
		RegressionEBKesch	11 hr - #30	N/A	2 hr 21 min
		RegressionEBSantis	20 hr - #7	N/A	1 hr 39 min

Do you want to know more?

- EasyBuild website: `http://hpcugent.github.io/easybuild`
- EasyBuild documentation: `http://easybuild.readthedocs.org`
- 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`