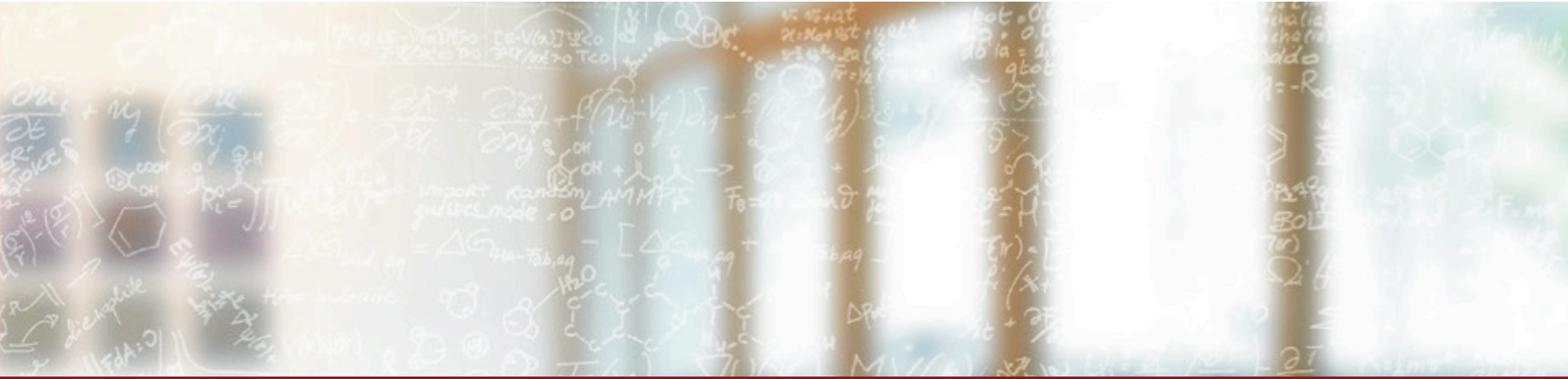




CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



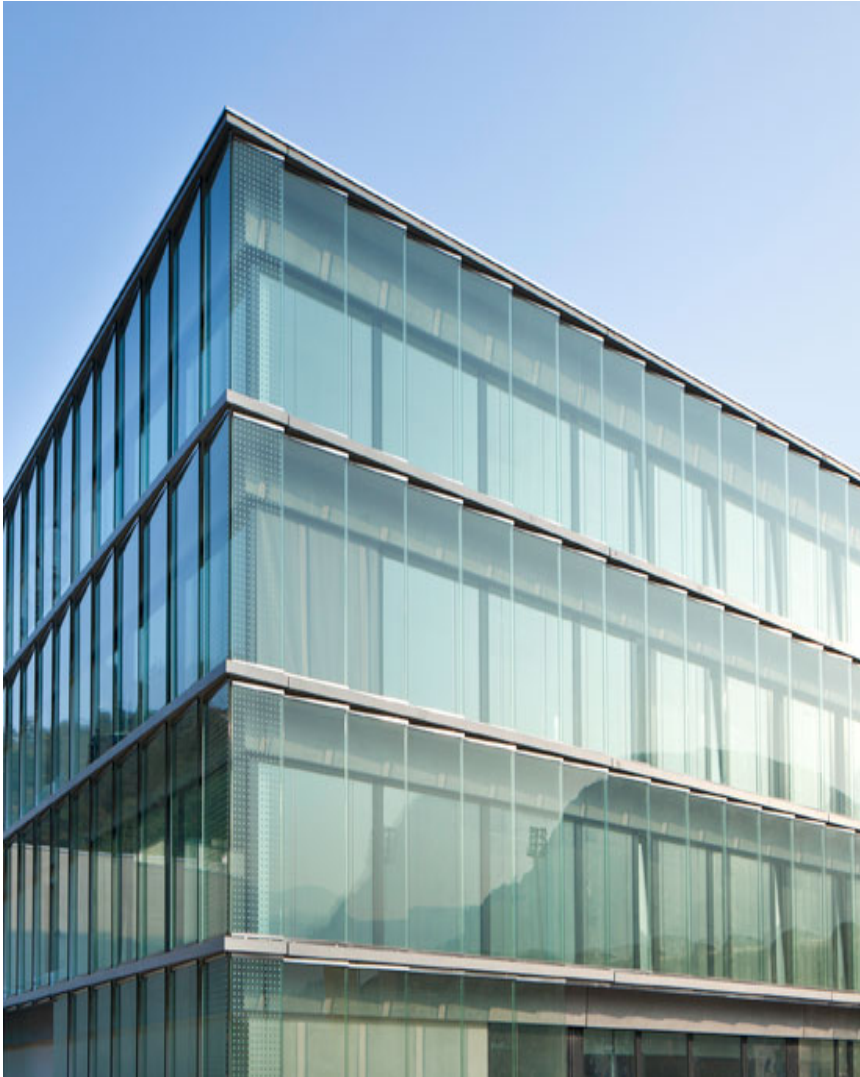
EasyBuild @ CSCS

2nd EasyBuild User Meeting

February 8th – 10th 2017, Juelich (Germany)

Luca Marsella and Guilherme Peretti-Pezzi - Scientific Computing Support (CSCS)

Outline



- **Why using EasyBuild?**
- EB + Cray Prog Env
 - External metadata / modules
- EB @ CSCS
 - Piz Kesch & Escha use case
 - Cray CS-Storm
 - Piz Daint use case
 - Cray XC
 - Github production repository
- Building software with Jenkins
- Conclusion

Why using EasyBuild?

- Lack of standard way to describe build recipes
 - Shell scripts, readme files, web/wiki pages, *invisible* docs
- Software available is very heterogeneous across systems
 - Moving users to a different machine requires a lot of work
- Systems upgrades are a huge overhead
 - Lots of manual work to re-deploy existing software
- Little collaboration with other sites doing the very same thing
 - Advantage to build a network to build HPC scientific applications

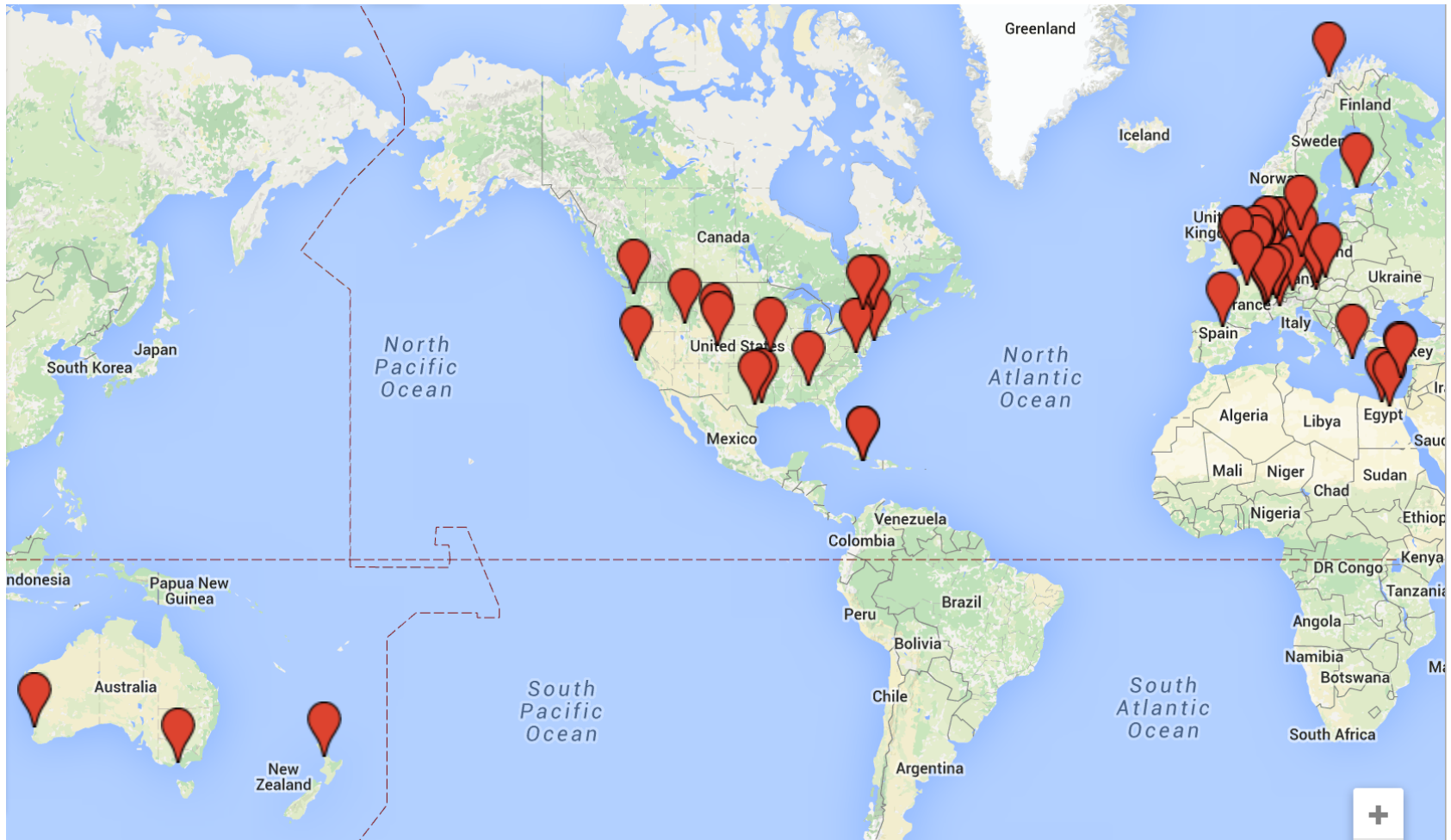
EasyBuild advantages

- open source python framework to **build** scientific software
- maintenance of **multiple software** deployments easy
- downloads, compiles and installs software packages
- **resolving dependencies** and **creating modulefiles** too
- adopted by many HPC centres, on **Cray systems** as well

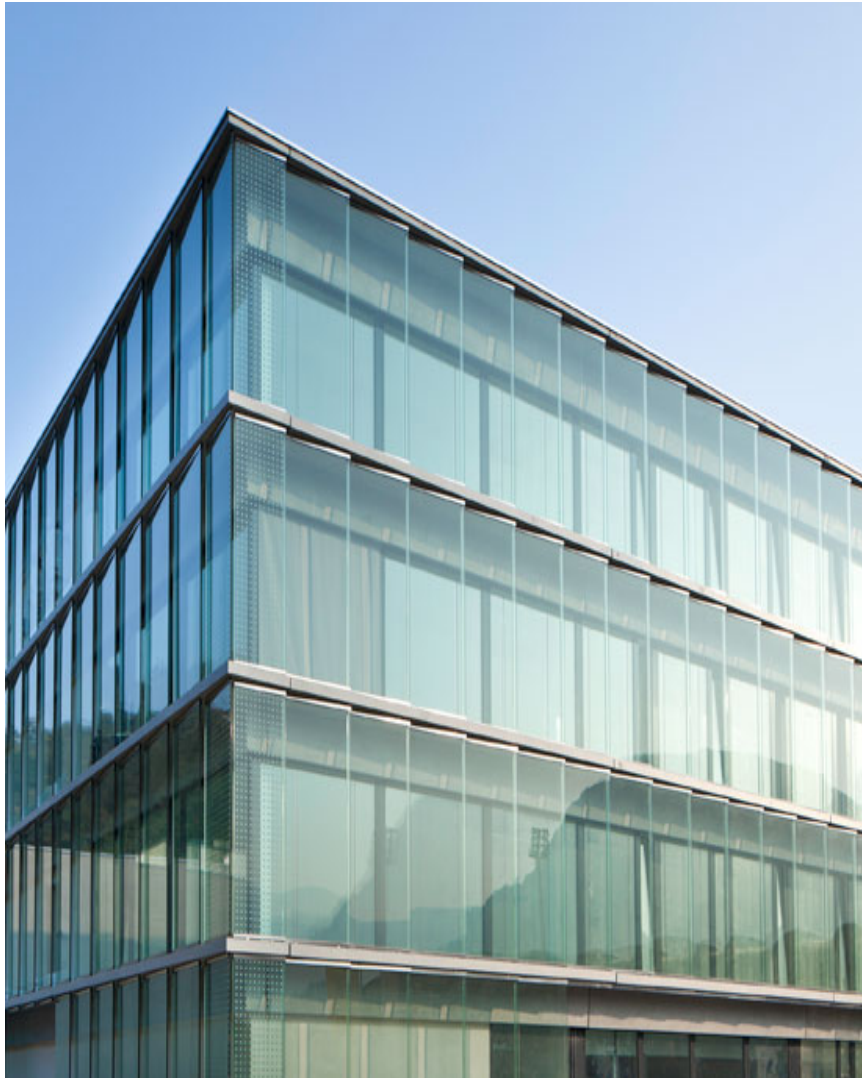
Local builds by users with EasyBuild

- `$ module load daint-gpu EasyBuild-custom`
- `$HOME/easybuild/<system-name>/<architecture>`
where `<architecture>` is either `<haswell>` or `<broadwell>`
 - `$ export EASYBUILD_PREFIX=/preferred/installation/folder`
 - `$ export EB_CUSTOM_REPOSITORY=/cscs/repository/folder`
 - `$ git clone https://github.com/eth-cscs/production.git`
- `$ module use $EASYBUILD_PREFIX/modules/all`
`$ module load <modulename>/version`
- `$HOME` folder is by default not readable by other users:
make builds available to each group with read-only access
- http://user.cscs.ch/compiling_and_optimizing/easybuild_framework/index.html

EasyBuild mailing list contributors



Outline



- Why using EasyBuild?
- **EB + Cray Prog Env**
 - **External metadata / modules**
- EB @ CSCS
 - Piz Kesch & Escha use case
 - Cray CS-Storm
 - Piz Daint use case
 - Cray XC
 - Github production repository
- Building software with Jenkins
- Conclusion

Software stack on a Cray XC

For example PE 2016.11 [1]

- Cray Compiling Environment - CCE 8.5.5
- Cray Message Passing Toolkit - MPT 7.5.0
- Perftools 6.4.3
- Cray Scientific and Math Libraries - CSML
 - LibSci 16.11.1
 - LibSci_ACC 16.11.1
 - PETSc 3.7.2.1
 - Trilinos 12.6.3.3
 - TPL 16.07.1
 - FFTW 3.3.4.10

CrayGNU-2016.11.eb

PrgEnv-gnu

- gcc/5.3.0
- cray-mpich/7.5.0
- cray-libsci/16.11.1
(BLAS, LAPACK,
ScaLAPACK, BLACS)

foss-2016.04.eb

- GCC/5.3.0
(binutils/2.26)
- OpenMPI/1.10.2
- OpenBLAS/0.2.18
- ScaLAPACK/2.0.2
(LAPACK-3.6.0)
- FFTW/3.3.4

- [1] <http://docs.cray.com/books/S-9408-1611/>

EasyBuild Enhancements for Cray Systems

1. Support for external module files
 2. Definition of Cray-specific toolchains
 3. Custom easyblock for Cray toolchains
- Various smaller enhancements specific to the Cray environment
 - **Thanks to Peter Forai & Kenneth Hoste**

1. Support for external module files

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

- EasyBuild can now 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) and 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.

2. Custom easyblock for Cray toolchains

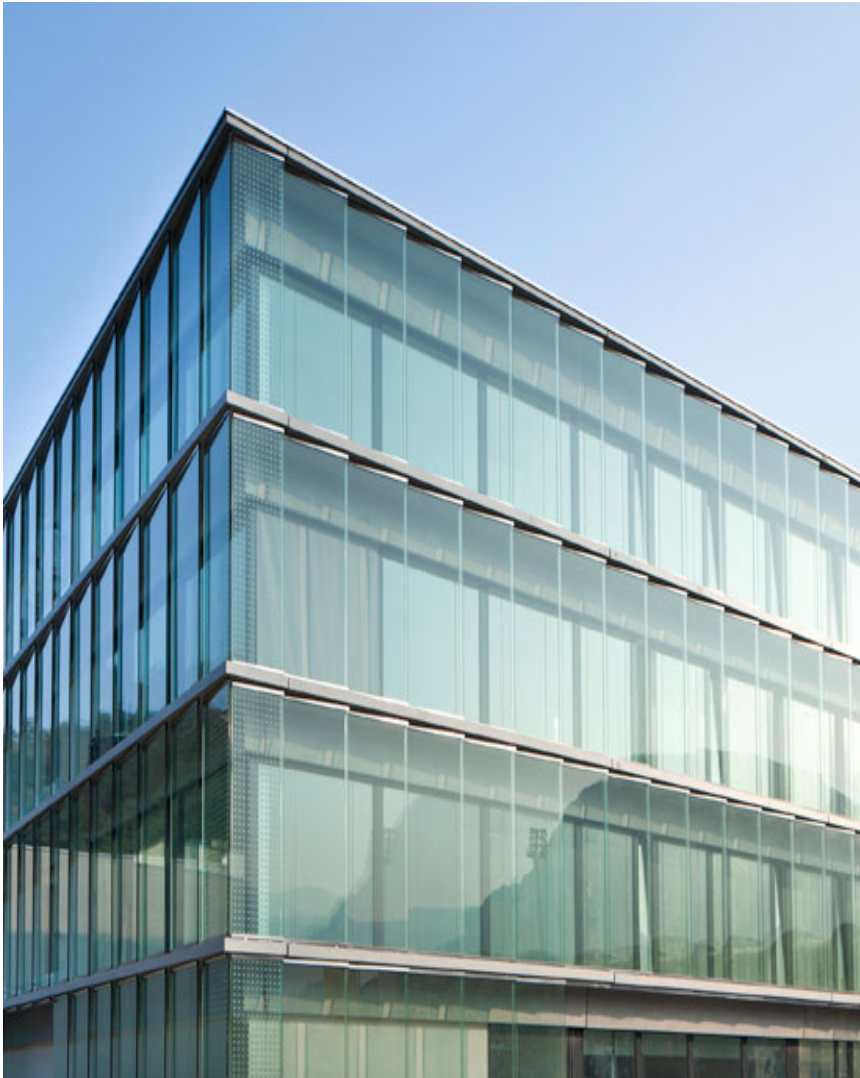
- This easyblock defines the **version pinned** components that make up the toolchain
- Easyblock implements logic to render the module files for the EasyBuild Cray toolchains
 - ensures that switching toolchain components works
 - avoids the need to run *module purge*
- New toolchain version combinations can then be placed in easyconfig file to ease creation of new toolchains.

3. Definition of Cray-specific Toolchains

Cray-specific toolchains have been implemented for each PrgEnv module:

- CrayCCE for PrgEnv-cray
 - CrayGNU for PrgEnv-gnu
 - CrayIntel for PrgEnv-intel
 - CrayPGI for PrgEnv-pgi
-
- Compiler component of the toolchains EasyBuild leverages the compiler wrappers provided by the Cray PE and EasyBuild exposes
 - `$CC`, `$CXX`, `$CFLAGS`, `$CXXFLAGS`, `$F77`

Outline



- Why using EasyBuild?
- EB + Cray Prog Env
 - External metadata / modules
- **EB @ CSCS**
 - **Piz Kesch & Escha use case**
 - Cray CS-Storm
 - **Piz Daint use case**
 - Cray XC
 - **Github production repository**
- Building software with Jenkins
- Conclusion

Overview of CSCS HPC systems

System	Scope	Accelerators / node	Type
Piz Daint	User Lab	1 GPU	Cray XC50
Monch	PASC projects	0	NEC Intel IvyBridge
Escha	Meteo Swiss	16 GPU	Cray CS-Storm
Kesch	Meteo Swiss	16 GPU	Cray CS-Storm
Leone	Large Memory	1 GPU	HP DL 360 Gen 9

Piz Kesch & Escha use case (MeteoSwiss / Cray CS-Storm)

“Kesch” and “Es-cha” consist of identical systems (production and failover), each comprising:

Cray CS-Storm: 12 nodes

- 2 x Intel Haswell E5-2690v3 2.6 GHz 12-core CPUs per node
 - total of 24 E5-2690v3 processors
- 256 GB 2133 MHz DDR4 memory per node
 - total of 3 TB
- 8 NVIDIA® Tesla® K80 GPU devices per node
 - total of 192 GPUs

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



Piz Kesch & Escha use case (MeteoSwiss / Cray CS-Storm)

- Cray PE is partially supported on the CS-Storm series:
 - PrgEnv-cray is available but not GNU or Intel
- System provided GCC-based compiler stack was unable to assemble optimized (AVX2) instructions for system's Intel Haswell processor.
 - GNU binutils was too old
- While waiting for a definitive fix from Cray...
 - The whole software stack required by MeteoSwiss was deployed using EasyBuild
 - Using a standard open source toolchain (gmvolff)
- EasyBuild software stack now in production since September/2015

Piz Daint

Model	Cray XC50/XC40
XC50 Compute Nodes	Intel® Xeon® E5-2690 v3 (Haswell) @ 2.60GHz (12 cores, 64GB RAM) and NVIDIA® Tesla® P100 16GB
XC40 Compute Nodes	Intel® Xeon® E5-2695 v4 (Broadwell) @ 2.10GHz (18 cores, 64/128 GB RAM)
Login Nodes	Intel® Xeon® CPU E5-2650 v3 @ 2.30GHz (10 cores, 256 GB RAM)
Interconnect Configuration	Aries routing and communications ASIC, and Dragonfly network topology
Scratch capacity	6.2 PB (Luster / Sonexion 3000)

Piz Daint



- #8 Top 500
 - #1 in Europe
 - 9779.0 PFLOPS
- #2 Green 500
 - 7453.5 MFLOPS/W

Available software on Cray using EasyBuild

▪ Stock EasyBuild repository

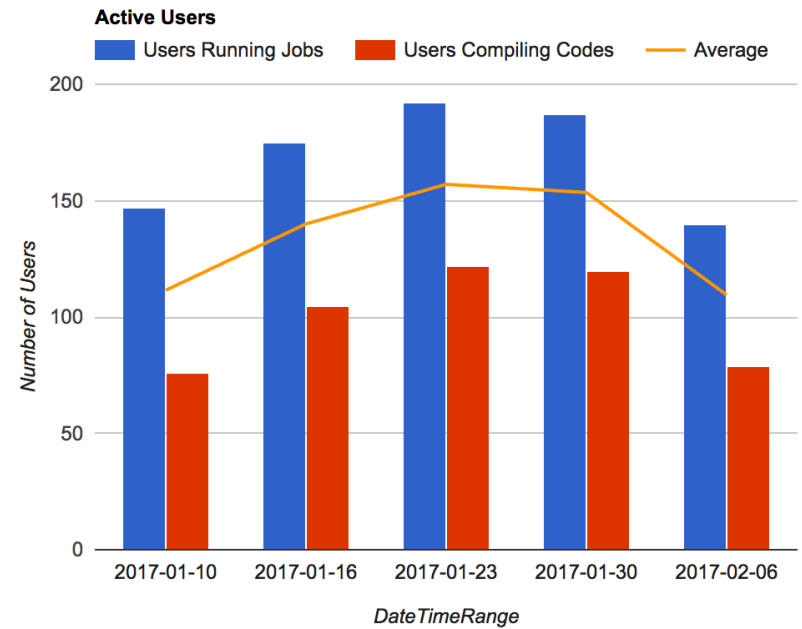
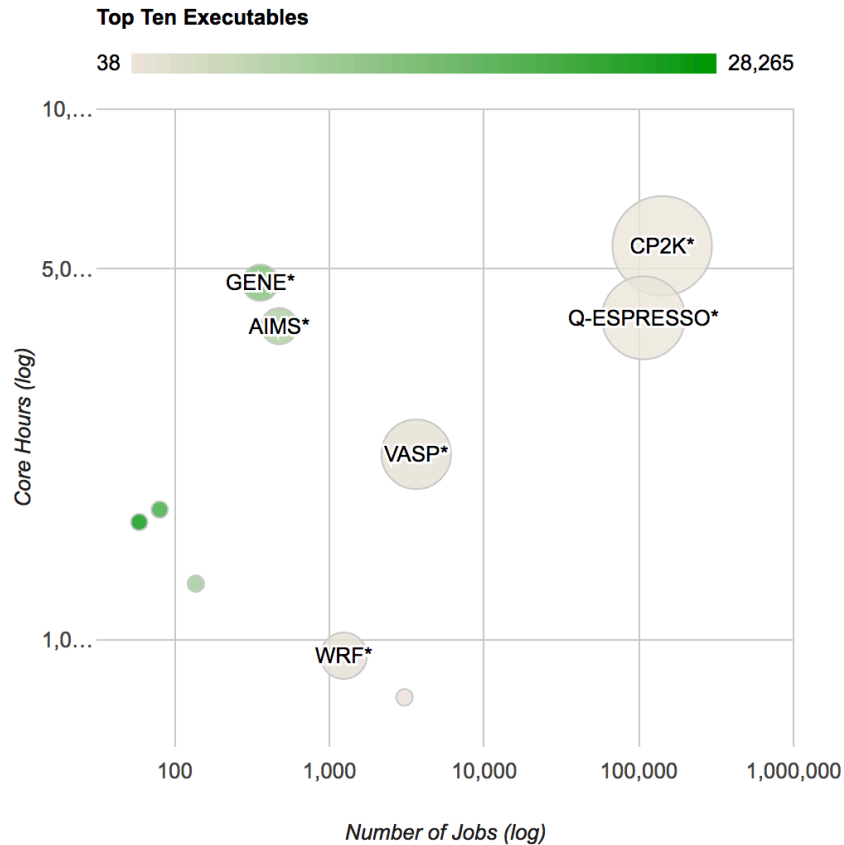
- Python, including
 - accelerated numpy + scipy, h5py, ...
- WRF
- CP2K[*]
- GROMACS[*]
- Boost
- GSL

▪ CSCS Production Github repository

- Amber[*]
- CDO
- CPMD
- LAMMPS[*]
- NCL
- NCO
- ParaView[*]
- Octave
- QuantumESPRESSO[*]
- R
- Scalasca
- ScoreP
- TensorFlow[*]
- VASP[*]
- Visit
- VMD[*]
- VTK[*]

[*] = GPU-enabled recipe available

Apps and Users using xalt statistics



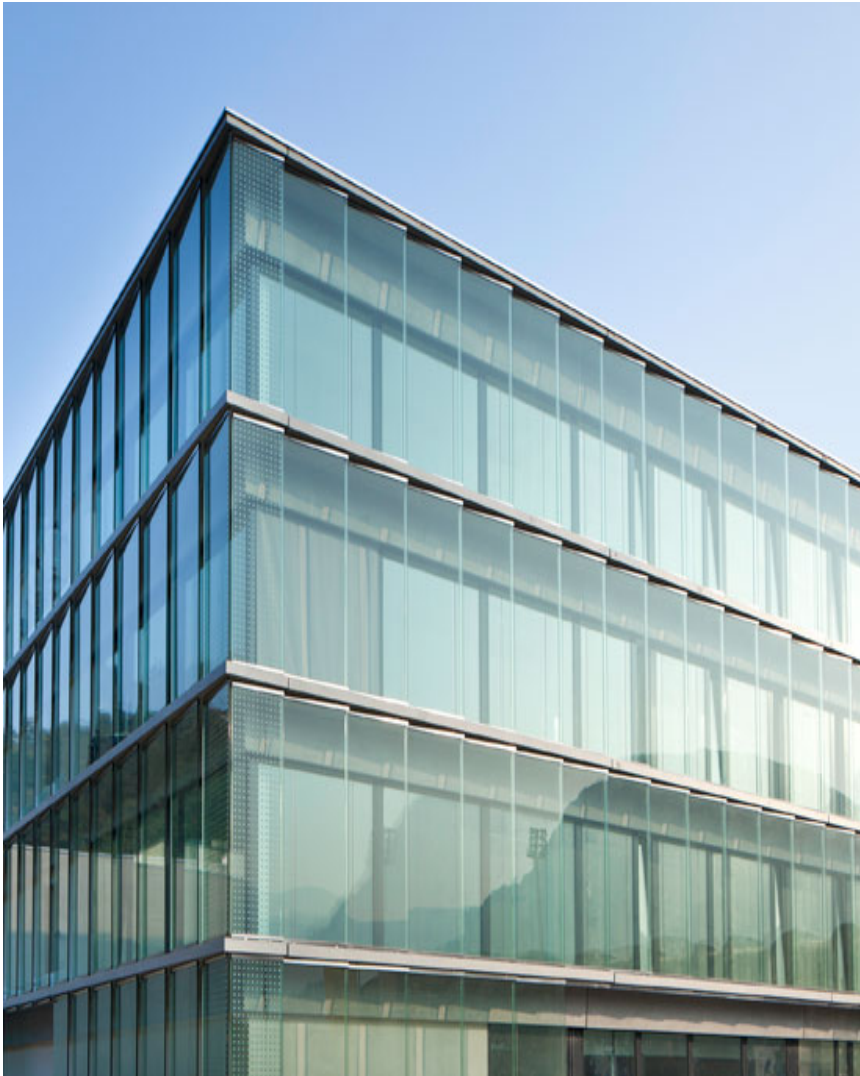
Github, EasyBuild & Continuous Integration

- Github repository hosting all recipes in production on Piz Daint
 - <https://github.com/eth-cscs/production>
 - Recipes go through a reviewing process (standard PR procedure)
- Automatic checking of build recipes on Piz Daint (by Jenkins)
 - GitHub Pull Request Builder Plugin for Jenkins
 - <https://github.com/janinko/ghprb>
 - Less error-prone & improved reproducibility
 - Robot ensures that recipes work without any extra tweaking
 - Such as exports and custom .bashrc files
- Autonomous deployment of software/modules on production (Jenkins)
 - List of easyconfig files is automatically deployed by Jenkins
 - For example, list of GPU-enabled software stack for the P100 partition
 - <https://github.com/eth-cscs/production/blob/master/jenkins-builds/6.0.UP02-2016.11-gpu>

Final comments: EasyBuild & Cray

- Proprietary and FOSS can co-exist
- Best of two worlds
 - Integrates new applications with optimized proprietary stack
 - Support is assured by Cray and also by the enthusiasts of the EB community
- Minimizes risks of vendor lock-in
 - EasyBuild provides alternatives in case of issues with software provided by Cray

Outline



- Why using EasyBuild?
- EB + Cray Prog Env
 - External metadata / modules
- EB @ CSCS
 - Piz Kesch & Escha use case
 - Cray CS-Storm
 - Piz Daint use case
 - Cray XC
 - Github production repository
- **Building software with Jenkins**
- Conclusion

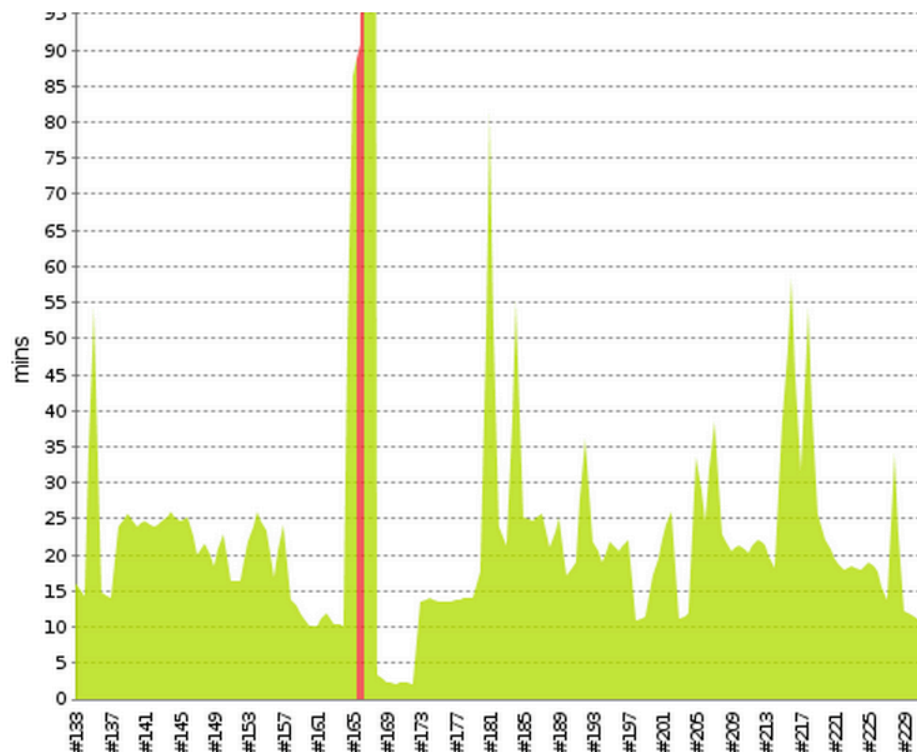
Jenkins

- Jenkins is used at CSCS for continuous integration/validation
- Advantages:
 - Several plugins are available
 - easily configured to run tasks by ssh anywhere
 - logs for all of your executions
 - info on past jobs and logs always accessible through the web interface
- Usage examples:
 - Development/Integration:
 - Checkout svn/git repositories to automatically build on different platforms
 - Validation
 - Periodically run unit tests
 - Monitoring
 - Periodically run sanity and performance tests (***regression***)
 - Run your favorite script or app
 - example at CSCS: driving the acceptance tests of new HPC systems

Monitoring the Lustre scratch performance with a cray-netcdf-hdf5parallel write test

Build Time Trend




Build	↑	Duration	Slave
#2		15 min	master
#3		16 min	master
#4		28 min	master
#5		30 min	master
#6		22 min	master
#7		20 min	master
#8		20 min	master
#9		20 min	master
#10		19 min	master
#11		17 min	master
#12		19 min	master
#13		18 min	master
#14		24 min	master
#15		18 min	master
#16		12 min	master
#17		11 min	master
#18		29 min	master
#19		39 min	master
#20		10 min	master

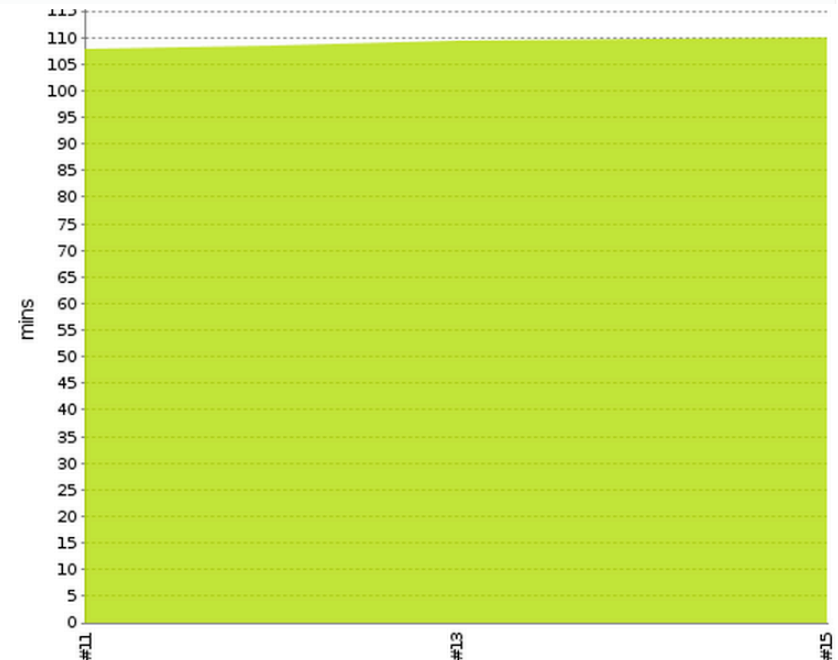


Building the EasyBuild software stack on Escha and Kesch (Meteo Swiss system)

S	W	Name ↓	Last Success	Last Failure	Last Duration
		RegressionEBKesch	20 hr - #15	N/A	1 hr 49 min 

Build Time Trend

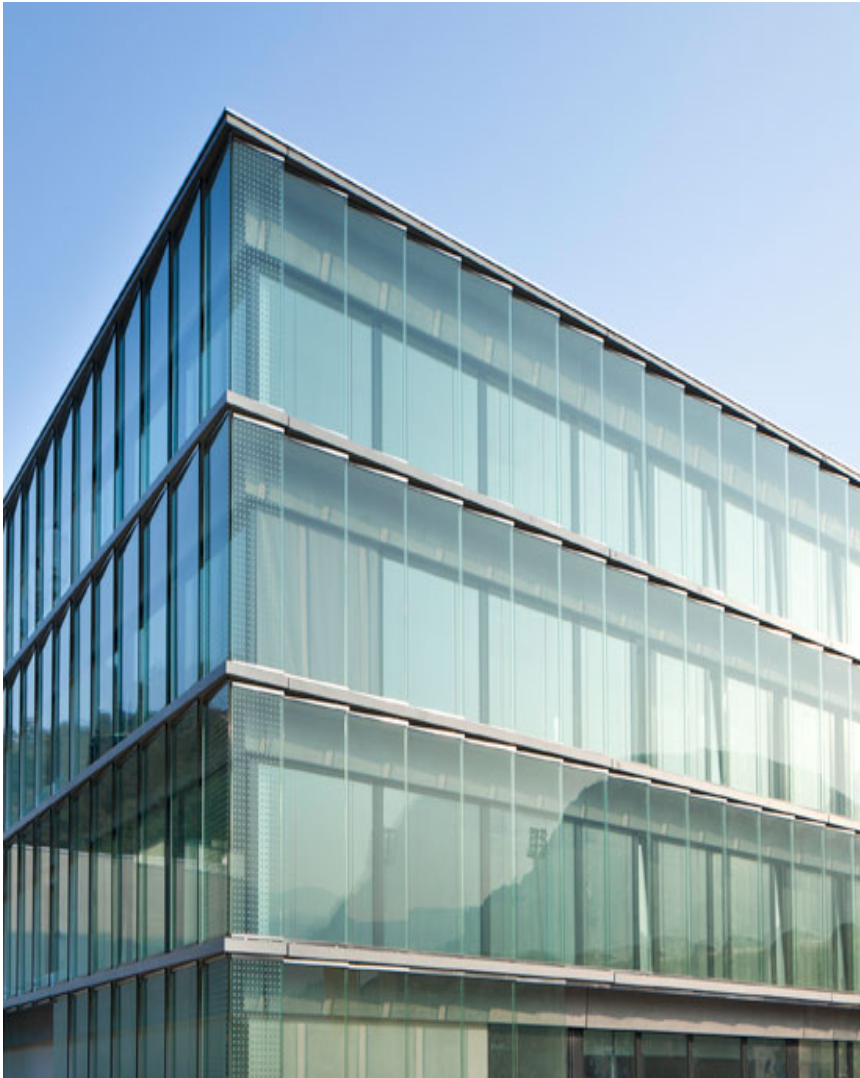
	Build ↑	Duration	Slave
	#11	1 hr 47 min	master
	#13	1 hr 49 min	master
	#15	1 hr 49 min	master



Jenkins with EasyBuild: workflow for recipes

- Testing new easyconfig files on systems with EasyBuild
- Workflow setup
 1. Create a folder accessible by jenscscs to store the .eb files
 - /path/to/eb-files/
 2. Create a jenkins project adding the target test system
 - CrayGNU = daint
 - foss/2015b = monch
 3. Add custom commands to the “Execute shell”
 - module load EasyBuild-custom
 - `find /path/to/eb-files/ -name '*CrayGNU-5.2.40*.eb' -exec eb {} "-r -f" \;`
- Usage
 1. Copy .eb files to /path/to/eb-files/
 2. Go to Jenkins and click on “Build now”

Outline



- Why using EasyBuild?
- EB + Cray Prog Env
 - External metadata / modules
- EB @ CSCS
 - Piz Kesch & Escha use case
 - Cray CS-Storm
 - Piz Daint use case
 - Cray XC
 - Github production repository
- Building software with Jenkins
- **Conclusion**

Conclusion

- Current EB installation is ready for application level
 - Validation with
 - Python : Piz Daint and Escha/Kesch
 - Escha/Kesch: complete software stack built with gmvolf toolchain
- Continuous validation techniques can be easily applied
 - Testing builds across all systems with Jenkins
 - weekly builds for every machine
 - Changes/errors on the PrgEnv can be detected early
- In order to get the most out of EasyBuild
 - We need to have consistent PrgEnv across
 - OK on Cray systems
 - Not currently true on non-Cray
 - Achievable with EasyBuild

Work in progress

- Stable Cray support:
 - <https://github.com/hpcugent/easybuild-framework/issues/1390>
- Rpath support to be tested on Cray systems
- Compatible build description with similar projects (Spack)
- Lower the bar for new users
 - For one build users need easyconfig + easyblock + framework
 - Extended-dry-run is currently the best approach

What can be improved?

- Implement new command line options for dependencies:
 - `--try-dep-version`
- Backup of custom easyblocks for reproducibility
- External modules:
 - Improve error reporting for missing modules
 - Generic/versionless entries on the metadata file
- Add more flexibility to the toolchain definition:
 - creating or modifying toolchains without changing the framework
 - Integration of EasyBuild with existing compilers
- Command line option to define default module version for builds

Useful links for EasyBuild @ CSCS

- EasyBuild @ CSCS Wiki on GitHub
 - <https://github.com/eth-cscs/production/wiki/User-instructions-for-EasyBuild>
- Easyconfig files repositories
 - List of production builds performed by Jenkins
 - <https://github.com/eth-cscs/production/tree/master/jenkins-builds>
 - Custom easyconfigs:
<https://github.com/eth-cscs/production/tree/master/easybuild/easyconfigs>
 - Custom easyblocks:
 - <https://github.com/eth-cscs/production/tree/master/easybuild/easyblocks>

Do you want to know more EasyBuild on Cray?

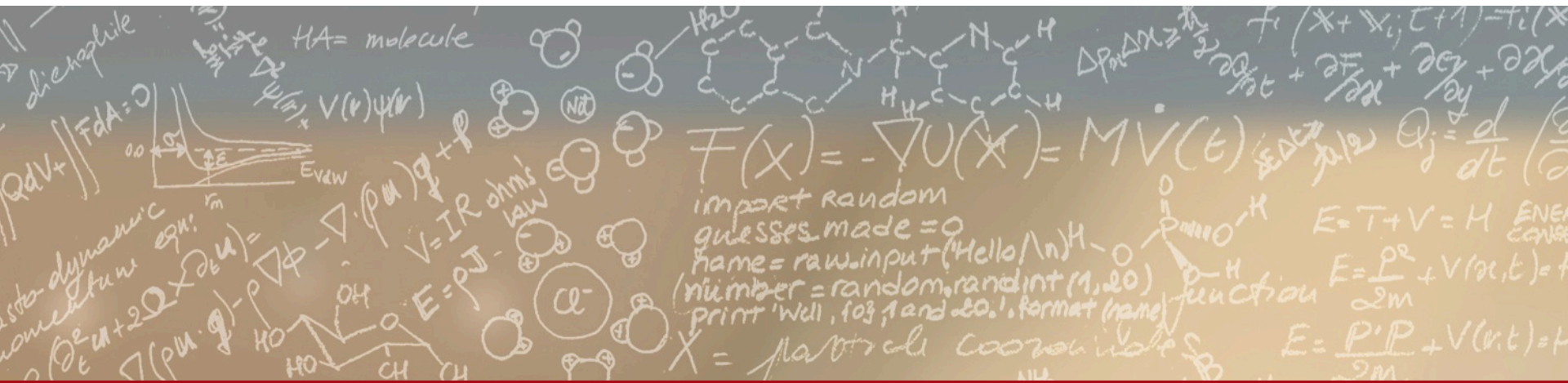
- Paper on the Cray User Group 2016
 - **Making Scientific Software Installation Reproducible On Cray Systems Using EasyBuild**
 - https://cug.org/proceedings/cug2016_proceedings/includes/files/pap145.pdf
- EasyBuild website: <http://hpcugent.github.io/easybuild>
- EasyBuild documentation: <http://easybuild.readthedocs.org>
- Stable EasyBuild releases: <http://pypi.python.org/pypi/easybuild>
- EasyBuild mailing list: easybuild@lists.ugent.be - <https://lists.ugent.be/wws/subscribe/easybuild>
- Twitter: http://twitter.com/easy_build
- IRC: #easybuild on chat.freenode.net



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Thank you for your kind attention