From EasyBuild to EESSI

Kenneth Hoste (HPC-UGent)

2022 Swiss Conference & HPCXXL User Group

9 March 2022 - virtual

whoami

kenneth.hoste@ugent.be

EasyBuild Slack

LinkedIn

@boegel (GitHub)

@kehoste (Twitter)

- Computer scientist with Masters + PhD from Ghent University
- Joined HPC-UGent team in October 2010
- Main tasks: user support, incl. installation of scientific software
- Inherited maintenance of EasyBuild in 2011
- Slowly also became EasyBuild lead developer & release manager ...
- I like beer, loud music, FOSS (Free & Open Source Software), dad jokes
- I don't like CMake, SCons, Bazel, TensorFlow, OpenFOAM, ...

in a nutshell

- **Framework to get scientific software installed** (preferably from source)
- Focus on Linux & HPC systems, specific attention to ***performance*** aspects
- Integrates with environment modules tool (Lmod, ...)
- Implemented in Python, supports Python 2.7 & 3.5+
- **GPLv2 open source license**, available via GitHub and Python Package Index (PyPI)
- Supports different compiler toolchains + over 2,600 software packages (excl. extensions)
- Created by HPC-UGent team, now supported & developed by a **worldwide community**...

Installing TensorFlow from source, with ease

https://easybuild.io

```
$ eb TensorFlow-2.6.0-foss-2021a-CUDA-11.3.1.eb
== temporary log file in case of crash /tmp/eb-GyvPHx/easybuild-U1TkEI.log
== processing EasyBuild easyconfig TensorFlow-2.6.0-foss-2021a-CUDA-11.3.1.eb
== building and installing TensorFlow/2.6.0-foss-2021a-CUDA-11.3.1...
== 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 /software/TensorFlow/2.6.0-...
== 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.
```

10+ years of EasyBuild

(April 2012)
First public release of EasyBuild
+ **start of EasyBuild community**

(Nov. 2012)
EasyBuild v1.0 (released *at* SC'12)

(mid-2009)
EasyBuild is created
by Stijn De Weirdt
(HPC-UGent tech lead)
as in-house project

in-house development

(Nov. 2014)
Proper documentation
easybuild.readthedocs.org

(2013 - 2019)
Getting Scientific Software Installed
SC/ISC Birds-of-a-Feather sessions

(March 2015)
EasyBuild v2.0
Backwards-incompatible changes

(2012 - 2016)
11 EasyBuild hackathons
across Europe (Univ. of Cyprus,
JSC, CSC.fi, CSCS) + TACC

(Nov. 2016)
EasyBuild v3.0
Lmod as default modules tool
Stable GitHub integration

(2015)
Integration with Cray PE
EasyBuild @ CSCS

(2016 - 2022)
7 EasyBuild User Meetings
(easybuild.io/eum**)**
@ Ghent, Jülich, Amsterdam,
Louvain-la-Neuve, Barcelona
+ 2x via Zoom/YouTube/Slack

(Sept. 2019)
EasyBuild v4.0
Compatible with Python 3
Only requires Python std. lib.

(Jan. 2021)
EasyBuild selected as
installation tool
for LUMI
(see EUM'22 talk)

(2020 - 2022)
EasyBuild tutorial at ISC
(easybuild.io/tutorial**)**
Online tutorial in June'20
Half-day tutorial @ ISC'21
Half-day tutorial @ ISC'22

High-level feature summary (1/2)

https://easybuild.io

- **Fully autonomously building and installing (scientific) software**
- Automatic generation of environment module files (Tcl or Lua syntax)
- Thorough logging of executed installation procedure
- Archiving of easyconfigs and patches
- **Highly configurable**, via configuration files + environment variables + command line
- Actively developed, frequent stable releases (every 6-8 weeks)

High-level feature summary (2/2)

https://easybuild.io

- Support for using a custom module naming scheme (incl. hierarchical)
- **Dynamically extendable** with additional easyblocks, toolchains, etc.
- **Transparency** via support for 'dry run' installation & trace output
- Extensively tested: unit tests, thorough testing of easyconfig PRs, regression testing
- **Integration with GitHub**, Slurm, FPM, Docker, Singularity, Rich, ...
- Worldwide collaboration between HPC sites small & large (JSC, CSCS, LUMI, ...)

Recent enhancements

https://easybuild.io

- Support for customizing EasyBuild through implementing **hooks**
- Automatic fallback for downloading source files from https://sources.easybuild.io
- **Easystack files** to facilitate managing of entire software stacks *[experimental]*
- **Adoption of FlexiBLAS in common `foss` toolchain**
- Improved support for RPATH links (see docs)
- **Integration with Rich**: colorful output, multi-level progress bars, ...
- Support for skipping extensions + **installing extensions in parallel** *[experimental]*
- Initial support for installing ROCm (AMD GPUs), NVHPC-based toolchains, RISC-V, ...

Site customization via hooks

https://easybuild.io

- **EasyBuild behaviour can be customized to adher to site policies by using hooks**
- Custom `*_hook` Python code to take action or manipulate internal data structures
- Different types of hooks: start/end hook, parse hook, pre/post-step hooks, ...
- Example: customizing configure options for Open MPI:

```
def pre_configure_hook(self, *args, **kwargs):

    if self.name == 'OpenMPI' and '--with-verbs' in self.cfg['configopts']:
        self.log.info("[pre-configure hook] Replacing --with-verbs with --without-verbs")
        self.cfg['configopts'] = self.cfg['configopts'].replace('--with-verbs', '--without-verbs')
```

- Extensively used by some EasyBuild sites (cfr. EUM'22 talk on heterogeneous MPI stack)
- More information: https://docs.easybuild.io/en/latest/Hooks.html

Easystack files

https://easybuild.io

- **Specifies software stack to install in YAML syntax**, supported since EasyBuild v4.3.2
- Primary key: software name, toolchain, software version(s)
- Enhancements to be implemented:
  - Allow specifying easyconfig file names
  - Support for specifying EasyBuild configuration settings
  - Filtering based on "labels" in EasyBuild configuration
  - Support for using regular expressions
- More information: https://docs.easybuild.io/en/latest/Easystack-files.html
- **Experimental feature** - requires configuring EasyBuild to enable using it - subject to change

example

```
software:
  GROMACS:
    toolchains:
      foss-2020a:
        versions: [2020.3]
      fosscuda-2020a:
        versions: [2020.3]
  OpenFOAM:
    toolchains:
      foss-2020a:
        versions: [8, v2006]
  R:
    toolchains:
      foss-2020a:
        versions: [4.0.0]
```

FlexiBLAS in `foss` toolchain

https://easybuild.io

- Link with applications with FlexiBLAS library rather than OpenBLAS, BLIS, MKL, ...
- **Select BLAS/LAPACK "backend" library to use at runtime, no need to re-link!**
- To use BLIS rather than OpenBLAS:

```
module load app/1.2.3-foss-2021b
module load BLIS/0.8.1-GCC-11.2.0
export FLEXIBLAS=BLIS
```

- Performance overhead is essentially non-existent (< 100 *clock cycles*)
- Also enables profiling to see which BLAS/LAPACK functions are most used

```
export FLEXIBLAS_HOOK=PROFILE
```

- See EasyBuild Tech Talk III on FlexiBLAS for more details
- MPItrampoline seems like a promising equivalent for MPI libraries...

Rich output, progress bars

https://easybuild.io

- **Multi-level progress bars** to show progress on downloading of files, installation steps, extensions, easyconfigs being installed, ...
- Requires having the  `rich_` Python package installed (which requires Python 3)

```
pip3 install easybuild rich
```

- Supported since EasyBuild 4.5.0, controlled via `--output-style` configuration setting
- More info: https://docs.easybuild.io/en/latest/Progress_bars.html

```
== fetching files...
Downloading gcc-11.2.0.tar.gz ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.4/143.2 MB 5.5 MB/s 0:00:17
Fetching files:  10% (1/10) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0:00:10 (gcc-11.2.0.tar.gz)
```

```
Installing 'reshape2' extension (116/990) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0:26:14
··· Installing R/4.1.0-foss-2021a: taking care of extensions (9 out of 17 steps done) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0:33:48
0:36:40  2 out of 3 easyconfigs done: CMake/3.20.1-GCCcore-10.3.0 (OK), bzip2/1.0.8-GCCcore-10.3.0 (OK)
```

Installing extensions in parallel

https://easybuild.io

- **Make EasyBuild leverage available cores to install multiple extensions concurrently**

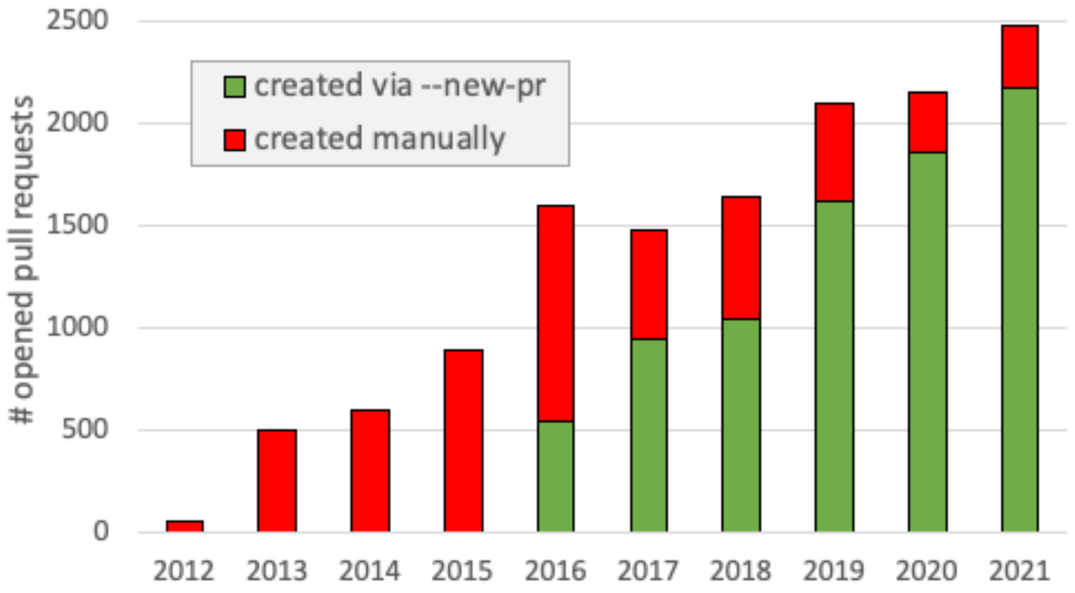
```
eb R.eb --parallel-extensions-install --experimental
```

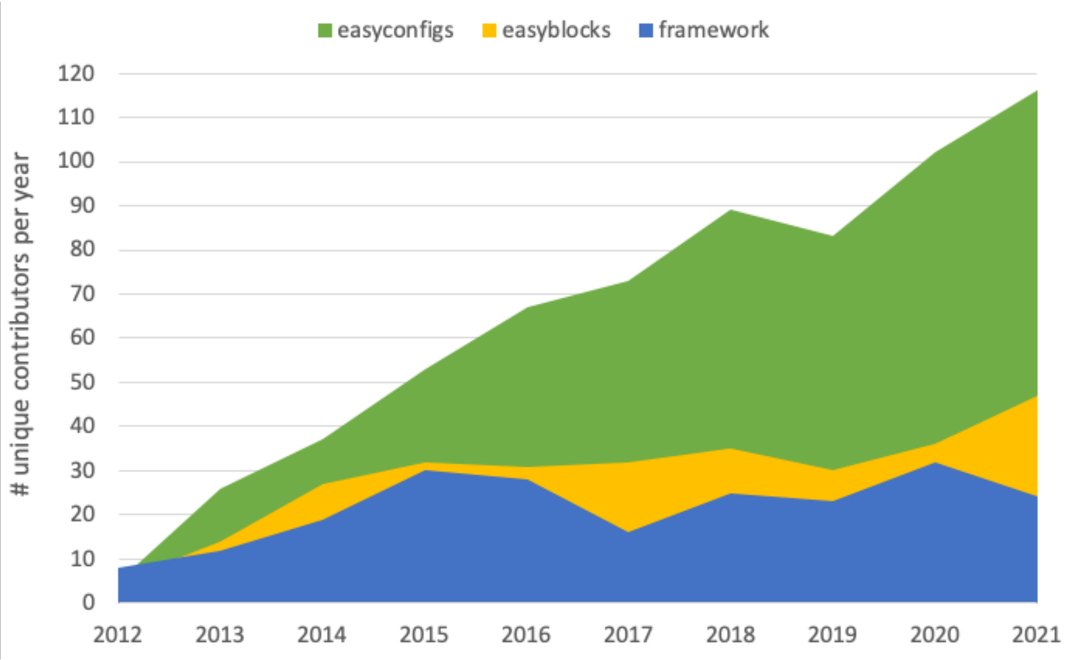
- Only works for R packages, extensions installed in dependencies not taken into account yet
- Significant speedup for R easyconfigs (which include over 1,000 extensions...)
- For other types of extensions (Python, ...) we need to known how to extract dependencies
- More info: https://docs.easybuild.io/en/latest/Installing_extensions_in_parallel.html
- **Experimental feature** - requires configuring EasyBuild to enable using it - subject to change

Community contributions

https://easybuild.io

- **Growing number of community contributions**: close to 2,500 easyconfig PRs per year!
- Over 100 unique contributors per year (and steadily increasing)
- Significant impact of GitHub integration features like `eb --new-pr`, see https://docs.easybuild.io/en/latest/Integration_with_GitHub.html

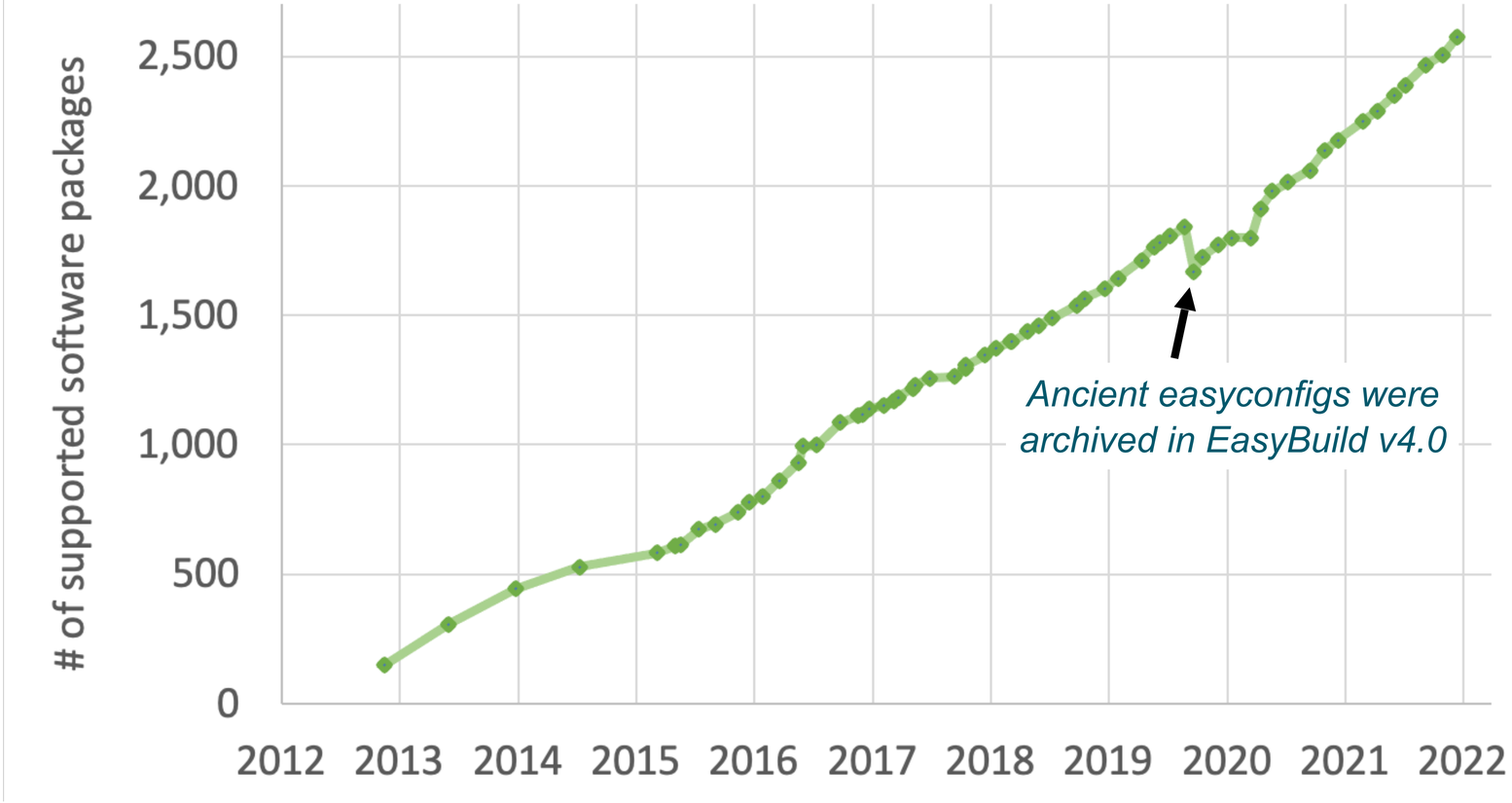




Community contributions

https://easybuild.io

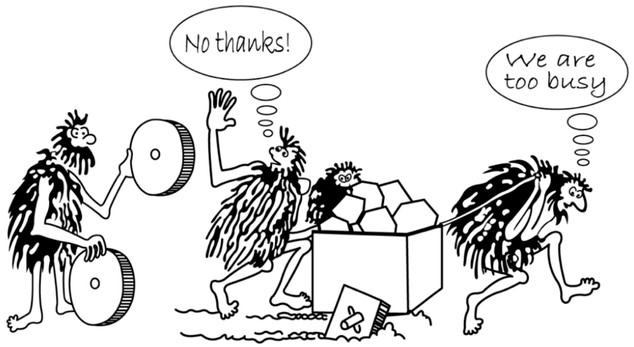
Evolution of number of supported software packages (excl. extensions)



Over 2,600 different software supported by EasyBuild v4.5.3 (plus ~2,000 extensions)

Stable upward trend, no sign of slowing down...

Landscape of computational science is changing



- **Explosion of available scientific software applications**
  - Broader adoption across scientific domains (bioinformatics, AI, ...)
  - Fueled by recent shifts in scientific community
    - Pressure to publish code along with research articles + popularity of GitHub
    - Wider adoption of HPC across scientific domains (GPUs in bioinformatics, ...)
- **Raised interest in using cloud infrastructure** (private and commercial)
- **Increasing variety in processor (micro)architectures**
  - Intel + AMD, ARM, POWER, soon also RISC-V?
  - In addition, GPUs (NVIDIA, AMD, soon Intel?) and other accelerators (TPUs, IPUs, ...)
- **In stark contrast: available manpower in HPC support teams...**

It is time to take the next step...

- EasyBuild helps a lot with managing scientific software stacks, but it's not enough anymore...
- **HPC sites *are still* losing way too much time in getting scientific software installed**
  - Installation requests for new software often require significant investment & expertise
  - Problems occur due to site-specific differences in system setup & configuration
  - Small details (OS packages, disk partitioning, ...) can cause trouble
- **Situation is getting *worse*, not better**: more software, increasing variety in hardware, ...
- **There is a lot of potential for more extensive collaboration beyond installing software**
  - Ensuring correctness of installations w.r.t. scientific results they produce
  - Evaluating performance of provided installations + performance monitoring
  - ReFrame developed by CSCS is definitely a step in the right direction!

Introducing EESSI...

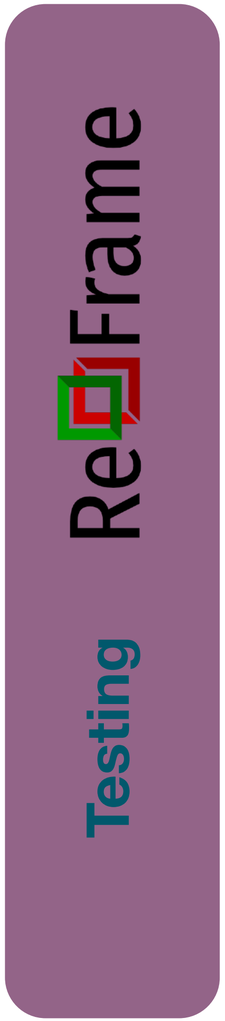
- *European Environment for Scientific Software Installations*
- Collaborative project, by and for the computational science community
- **Shared central stack of (optimized) scientific software installations**
- Uniform way of providing software to users, regardless of system they use
- Should work regardless of OS and system architecture (HPC, cloud, ...)
- Focus on **performance**, automation, testing, collaboration

EESSI
EUROPEAN ENVIRONMENT FOR
SCIENTIFIC SOFTWARE INSTALLATIONS

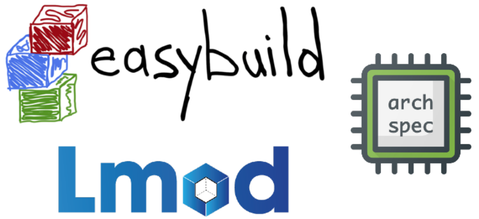
https://www.eessi-hpc.org

https://eessi.github.io/docs (try out pilot setup!)

High-level overview of EESSI



Software layer
applications + dependencies



Compatibility layer
levelling the ground across Linux distros

Filesystem layer
distribution of the software stack

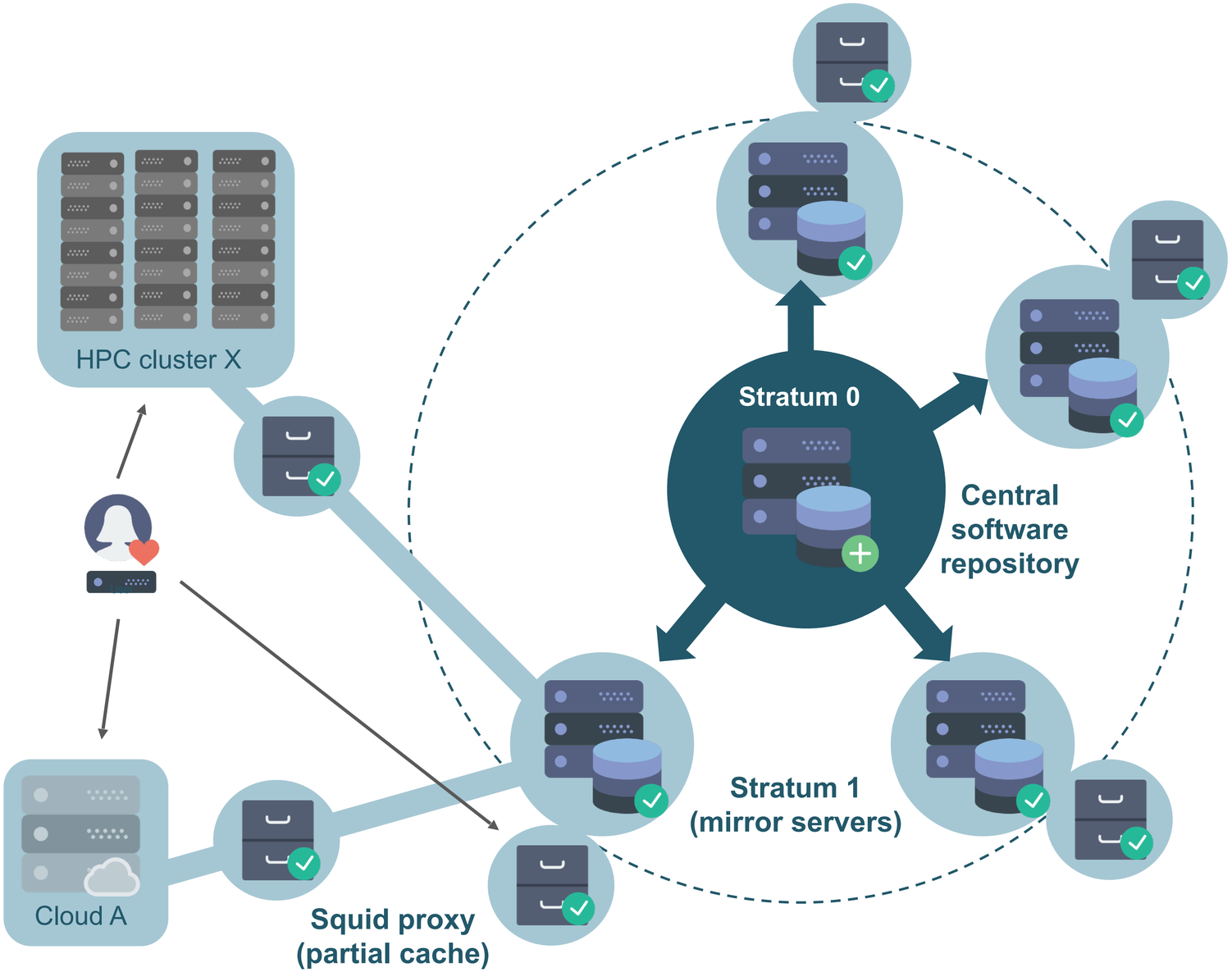
Host OS provides network & GPU drivers, resource manager (Slurm), ...

host operating system (any Linux distribution)

Heavily inspired by

software stack

Software distribution via CernVM-FS



CernVM-FS

https://cvmfs.readthedocs.io

- Global distribution of software installations
- Software is download *on-demand* over HTTP
- Transparent for end users, feels like local software
- Added software is *automatically* picked up
- Redundant network of “mirror” servers
- Multiple levels of caching (start-up performance)
- **Same software stack everywhere**:
  laptops, workstations, HPC clusters, cloud VMs, …

(icons via https://www.flaticon.com/authors/smashicons)

From zero to science in 3 steps

- Step 1: Get access to EESSI repository, either through:
  - system-wide CernVM-FS installation (requires admin privileges)
  - container with CernVM-FS + EESSI configuration pre-installed
- Step 2: Set up environment: source EESSI init script
- Step 3: Load module(s) and run!

https://eessi.github.io/docs/pilot

https://github.com/eessi/eessi-demo

```
# Assumed status: EESSI is accessible (CernVM-FS installed + EESSI configuration in place)

# Step 2: set up environment (CPU architecture is detected automatically)
$ source /cvmfs/pilot.eessi-hpc.org/latest/init/bash

# Step 3: load module(s) to activate software (check with 'module avail'), and run!
[EESSI pilot 2021.06] $ module load GROMACS
[EESSI pilot 2021.06] $ gmx mdrun ...
```

Use cases

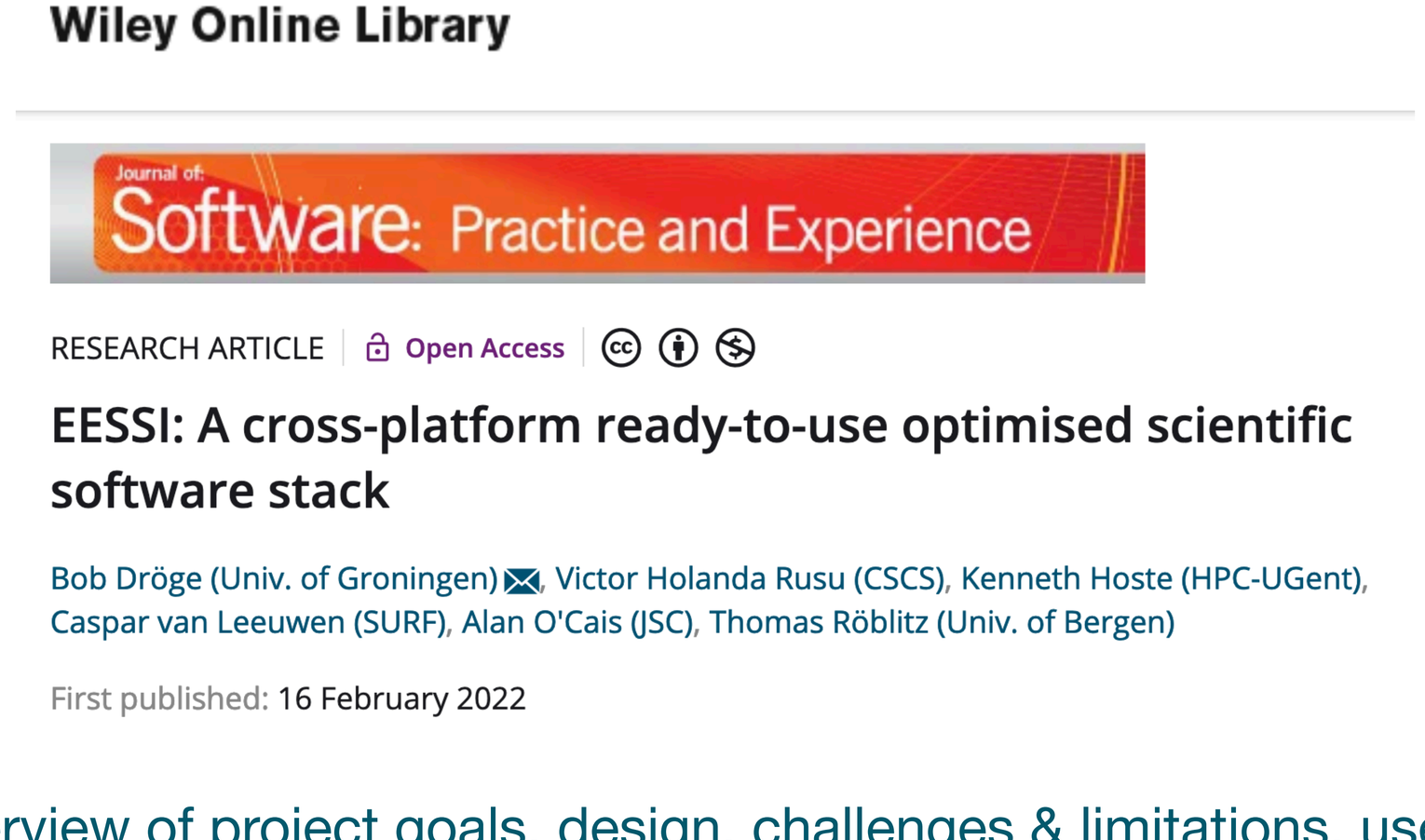
- A **uniform software stack** across HPC clusters, clouds, servers and laptops
- Can be leveraged in **continuous integration** (CI) environments for software testing
- Significantly facilitates setting up **infrastructure for HPC training**
    - Using throwaway clusters in the cloud
    - Participants can easily get access to same software environment "at home"
- **Enhanced collaboration** with software developers and application experts
    - Work towards "vetting" of scientific software installations included in EESSI
    - Relieve developers from burden of getting their software (properly) installed
- **Community software** and **portable workflows**

Current status

- Started as an idea in Feb'20, effort kickstarted that same year
- Monthly online meetings since mid 2020
- Currently an unfunded "side project", actively pursuing dedicated project funding...
- **Work-in-progress, proof-of-concept pilot repository available - eessi.github.io/docs/pilot**
  - Supports `x86_64` (Intel, AMD), `aarch64` (Arm 64-bit), `ppc64le` (POWER)
  - Included software: Bioconductor, GROMACS, OpenFOAM, R, TensorFlow, ...
  - Targets: Intel Haswell + Skylake, AMD Rome + Milan, AWS Graviton2, POWER9 (partial)
- Significant amount of sponsored credits provided by both AWS and Azure
- Actively developing monitoring, tooling, test suite, GPU support, contribution workflow, etc.

EESSI paper (open access)

https://doi.org/10.1002/spe.3075



Wiley Online Library

Journal of: Software: Practice and Experience

RESEARCH ARTICLE | Open Access

EESSI: A cross-platform ready-to-use optimised scientific software stack

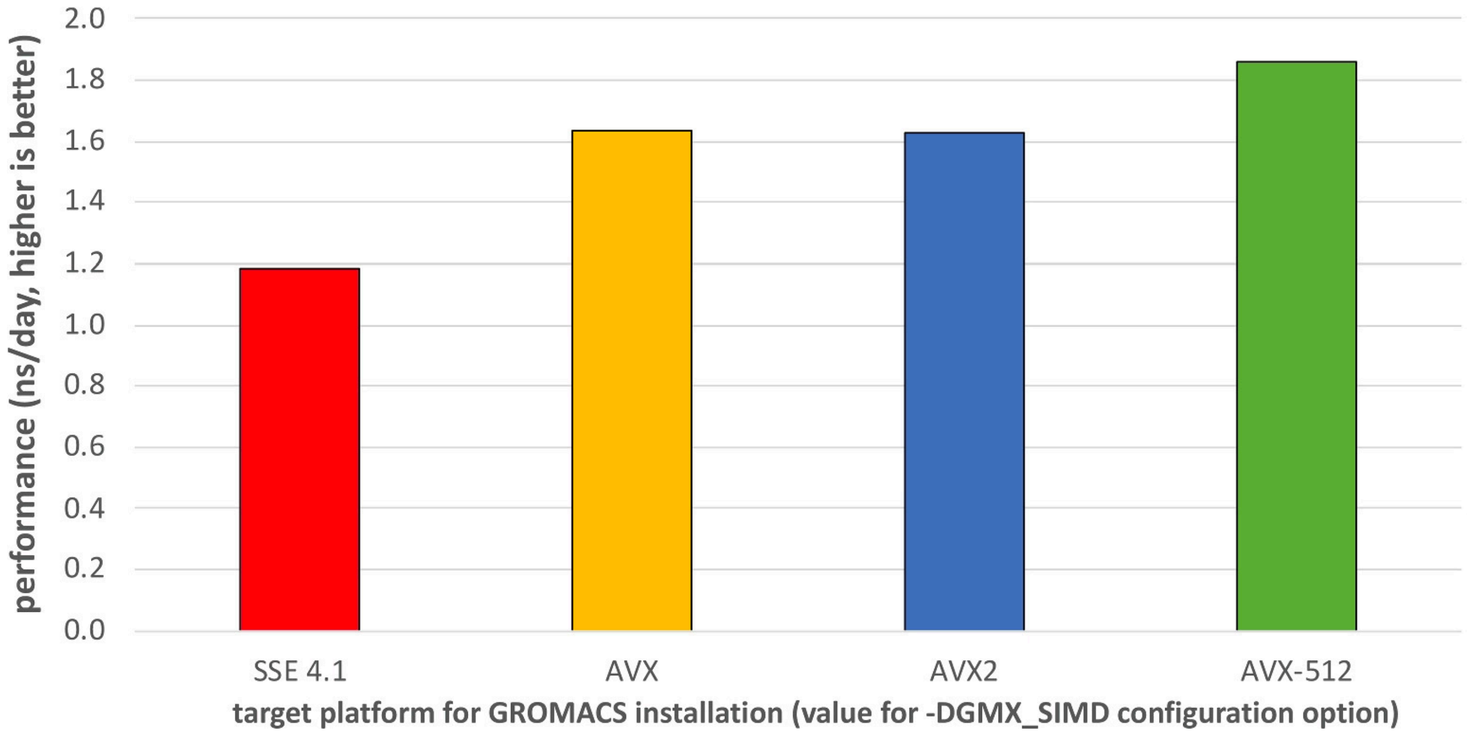
Bob Dröge (Univ. of Groningen), Victor Holanda Rusu (CSCS), Kenneth Hoste (HPC-UGent), Caspar van Leeuwen (SURF), Alan O'Cais (JSC), Thomas Röblitz (Univ. of Bergen)

First published: 16 February 2022

Detailed overview of project goals, design, challenges & limitations, use cases, etc.

EESSI paper (open access)

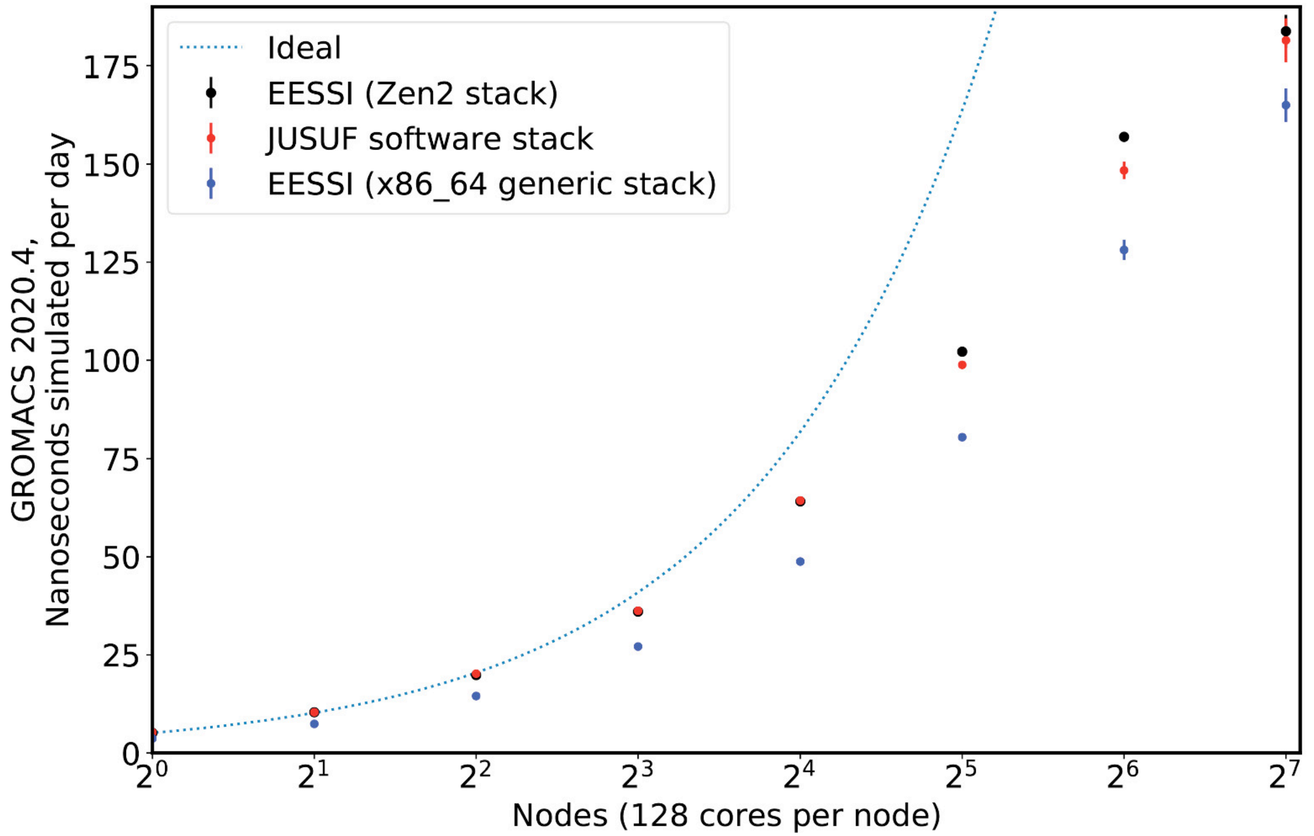
https://doi.org/10.1002/spe.3075



- Motivation for EESSI: keeping the 'P' in HPC (also when using cloud or containers)
- Impact of target instruction set for GROMACS: 57% speedup on CPUs that support AVX-512
- When using containers, performance is often sacrificed for "mobility of compute"...

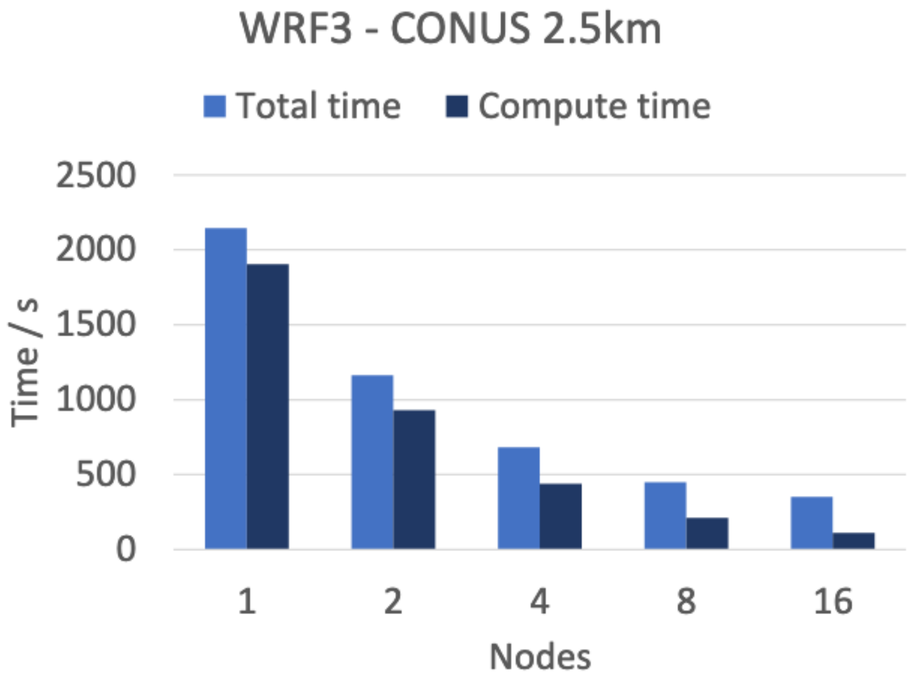
EESSI paper (open access)

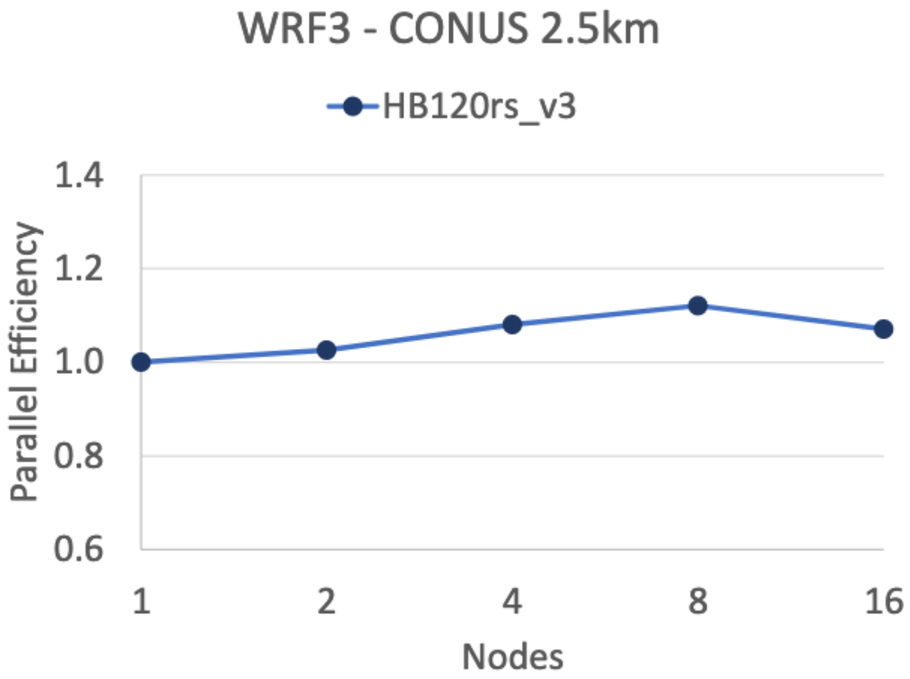
https://doi.org/10.1002/spe.3075



- Includes proof-of-concept performance evaluation compared to system software stack
- Performed at JUSUF @ JSC using GROMACS 2020.4, up to 16,384 cores (CPU-only)

Running WRF in Azure via EESSI





- Performance experiment by Hugo Meiland & Davide Vanzo (Microsoft Azure),
  for their EUM'22 talk "Leveraging EESSI for WRF simulations at scale on Azure HPC"
- WRF v3.9.1.1 with `foss/2020a` on `HB120rs_v3` (120 cores, AMD EPYC 7V13, HDR Infiniband)
- Essentially using EESSI pilot repository as is, no changes needed to leverage fast interconnect!

EESSI

EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

eessi-hpc.org

eessi.github.io/docs

github.com/eessi

Join our mailing list & Slack channel

eessi-hpc.org/join

Twitter: @eessi_hpc

Status page: status.eessi-infra.org

Paper: https://doi.org/10.1002/spe.3075

EESSI-related talks at EUM'22:

Getting started - WRF in Azure - Adding software

Monthly online meetings, open to anyone interested

(first Thursday, 14:00 CE(S)T)