EESSI

EUROPEAN ENVIRONMENT FOR
SCIENTIFIC SOFTWARE INSTALLATIONS

EESSI & CernVM-FS meeting

Sept 18th 2020

Outline:

- Introductions (who's who)
- EESSI in a nutshell
- CernVM-FS in EESSI + current status
- Building software for EESSI: goals & questions
- More questions

EESSI
EUROPEAN ENVIRONMENT FOR
SCIENTIFIC SOFTWARE INSTALLATIONS

Quick introduction

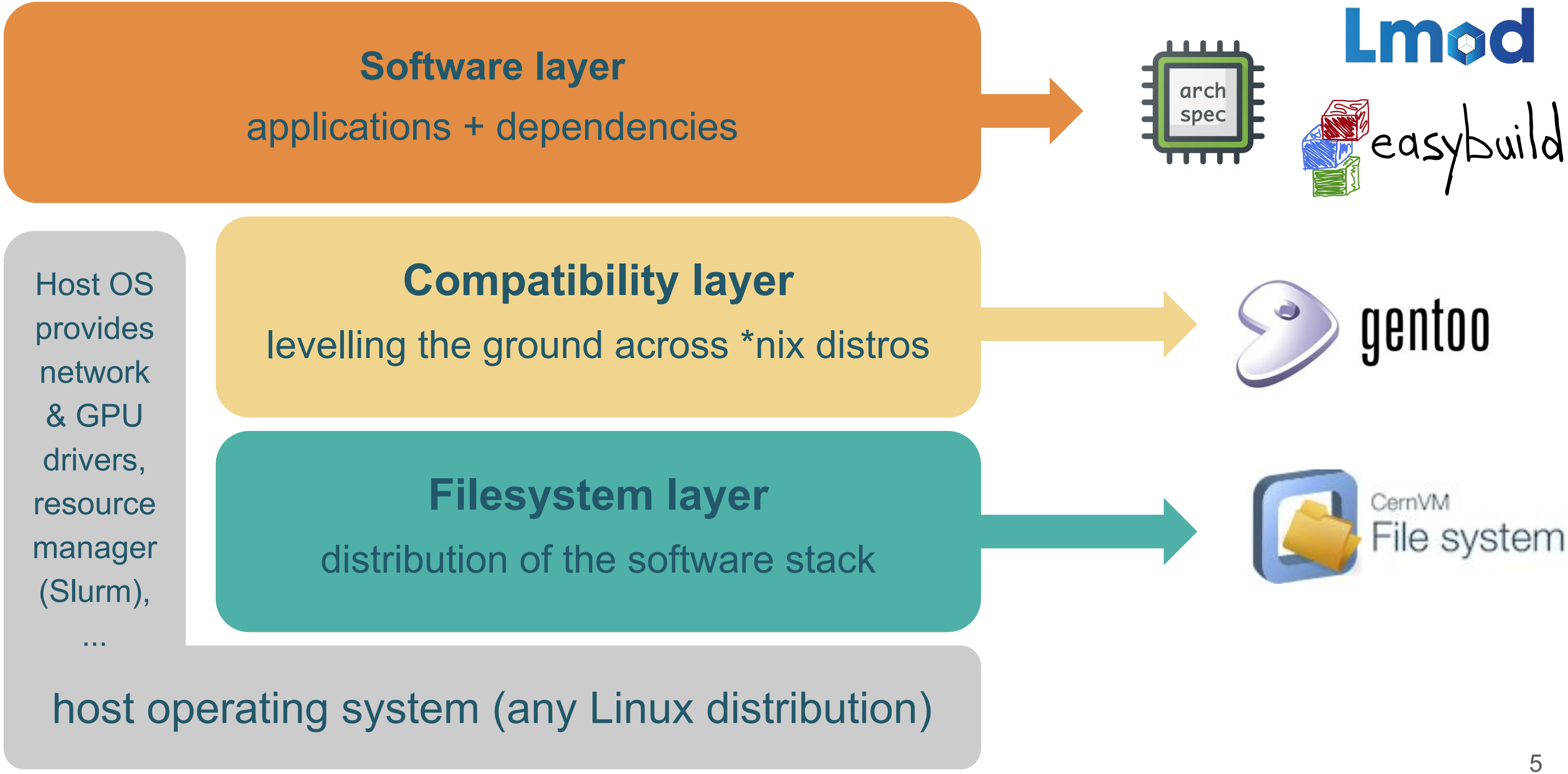
- European Environment for Scientific Software Installations (EESSI, pronounced as “easy”)
- Collaboration between different European partners in HPC community
- Goal: building a **common scientific software stack for HPC systems & beyond**
- *Heavily* inspired by work done by Compute Canada
- “Grass roots” project, fueled by a lack of time to do a proper job at installing scientific software and the desire for collaborating on something useful (+ having beers together)
  - => No dedicated funding/manpower yet

Scope & goals

- Shared repository of (optimized!) scientific software installations
- Avoid duplicate work across HPC sites
- Uniform way of offering software to users, regardless of system they use
- Should work on any (common) *nix distribution and system architecture
  - From laptops and personal workstations to HPC clusters and cloud
  - Support for different CPUs, interconnects, GPUs, etc.
- Focus on **performance**, automation, testing, collaboration

EESSI
EUROPEAN ENVIRONMENT FOR
SCIENTIFIC SOFTWARE INSTALLATIONS

High-level overview of the EESSI project



Software layer

applications + dependencies

Compatibility layer

levelling the ground across *nix distros

Filesystem layer

distribution of the software stack

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

host operating system (any Linux distribution)

Keeping the P in HPC

- Software should be optimized for specific system architectures (CPU, GPU)
- Impact on performance is often significant for scientific software
- Example: GROMACS

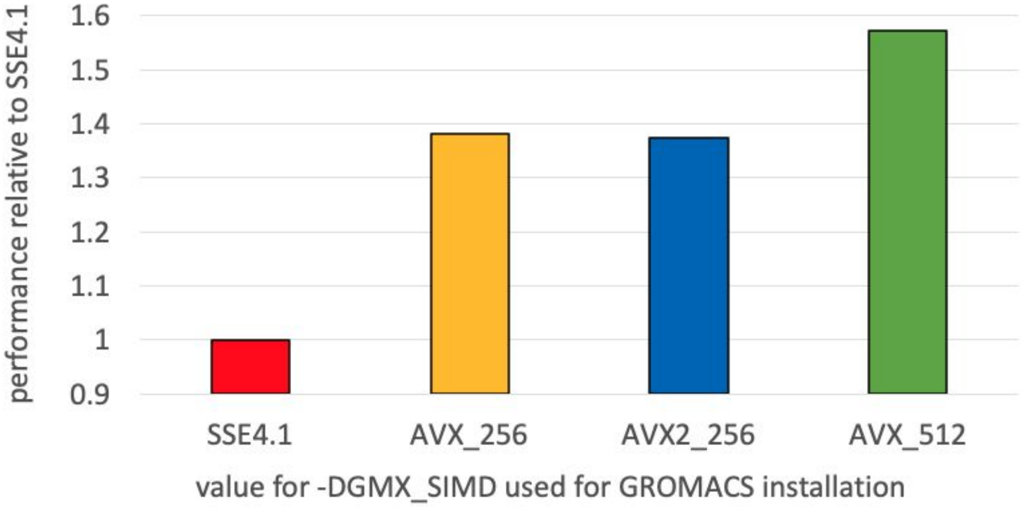
GROMACS 2020

compiled with foss/2020a (GCC 9.3, FFTW 3.3.8)
EasyBuild v4.2.2

2x Intel Xeon Gold 6420 (Cascade Lake)
CentOS 7.8

36 threads (no MPI)

Test Case B from PRACE-UEABS
(https://repository.prace-ri.eu/git/UEABS/ueabs)



CernVM-FS in EESSI

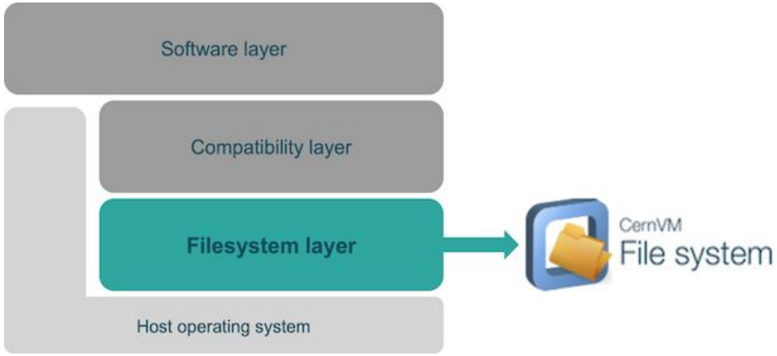
- Transport layer for software in the compatibility and software layers
- One subtree per system architecture (CPU, GPU)
  - Clients automatically use the best suited subtree based host CPU, GPU, ...
- Config repository for distributing configuration files
- Different repositories for testing and production (+ another for licensed software?)
  - Currently only one “pilot” repository
- Planning to use GEO API for finding the closest Stratum 1

Current status: EESSI pilot repository

- Stratum-0 + one single Stratum-1 with CernVM-FS 2.7.4 (hosted at rug.nl)
- Singularity container for clients with CernVM-FS 2.7.2 (very easy to use)
- First version (2020.08) of pilot repository:
  - Only installations for Intel Haswell (but compatible with any (?) client OS!)
  - Toolchain: GCC 9.3.0 + Open MPI 4.0.3 + OpenBLAS 0.3.9 + FFTW 3.3.8
  - Software: Python, GROMACS, OpenFOAM, ParaView, … (+ all deps)
  - Easy to use init script to set up environment (hardcoded to intel/haswell)
  - Various known shortcomings (Open MPI config, `$LD_LIBRARY_PATH`, etc.)
- More details at https://eessi.github.io/docs/pilot

EESSI directory structure (pilot)

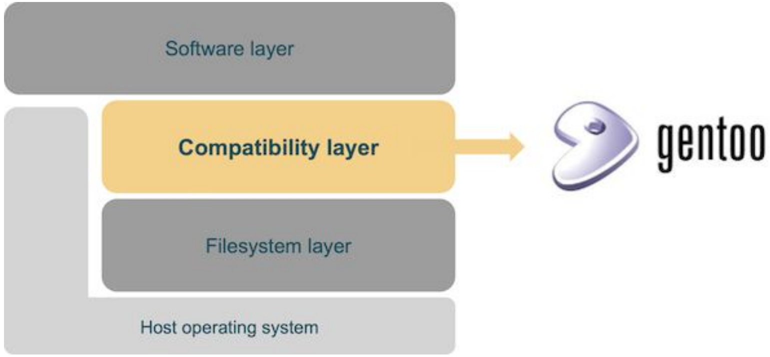
```
.
└── cvmfs/
    ├── cvmfs-config.eessi-hpc.org
    └── pilot.eessi-hpc.org/
        └── 2020.08/
            ├── compat/
            │   ├── aarch64
            │   ├── ppc64le
            │   └── x86_64
            └── software/
                ├── aarch64
                ├── ppc64le
                └── x86_64
```



(see https://eessi.github.io/docs/pilot)

EESSI directory structure (pilot)

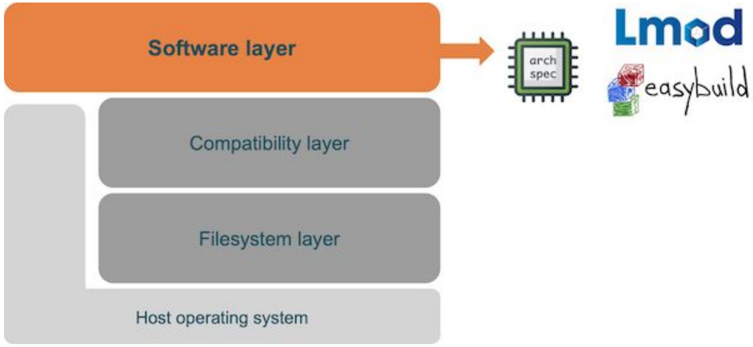
```
.
└── cvmfs/
    ├── cvmfs-config.eessi-hpc.org
    └── pilot.eessi-hpc.org/
        └── 2020.08/
            ├── compat/
            │   ├── aarch64
            │   ├── ppc64le
            │   └── x86_64/
            │       ├── bin
            │       ├── etc
            │       ├── lib64
            │       ├── sbin
            │       └── usr
            └── software/
                ├── aarch64
                ├── ppc64le
                └── x86_64
```



(see https://eessi.github.io/docs/pilot)

EESSI directory structure (pilot)

```
.
└── cvmfs/
    ├── cvmfs-config.eessi-hpc.org
    └── pilot.eessi-hpc.org/
        └── 2020.08/
            ├── compat
            └── software/
                ├── aarch64
                ├── ppc64le
                └── x86_64/
                    ├── amd/
                    │   └── zen2
                    └── intel/
                        ├── skylake
                        └── haswell/
                            ├── modules
                            └── software/
                                ├── GROMACS/
                                │   └── 2020.1-foss-2020a-Python-3.8.2
                                └── OpenFOAM/
                                    ├── 8-foss-2020a
                                    └── v2006-foss-2020a
```

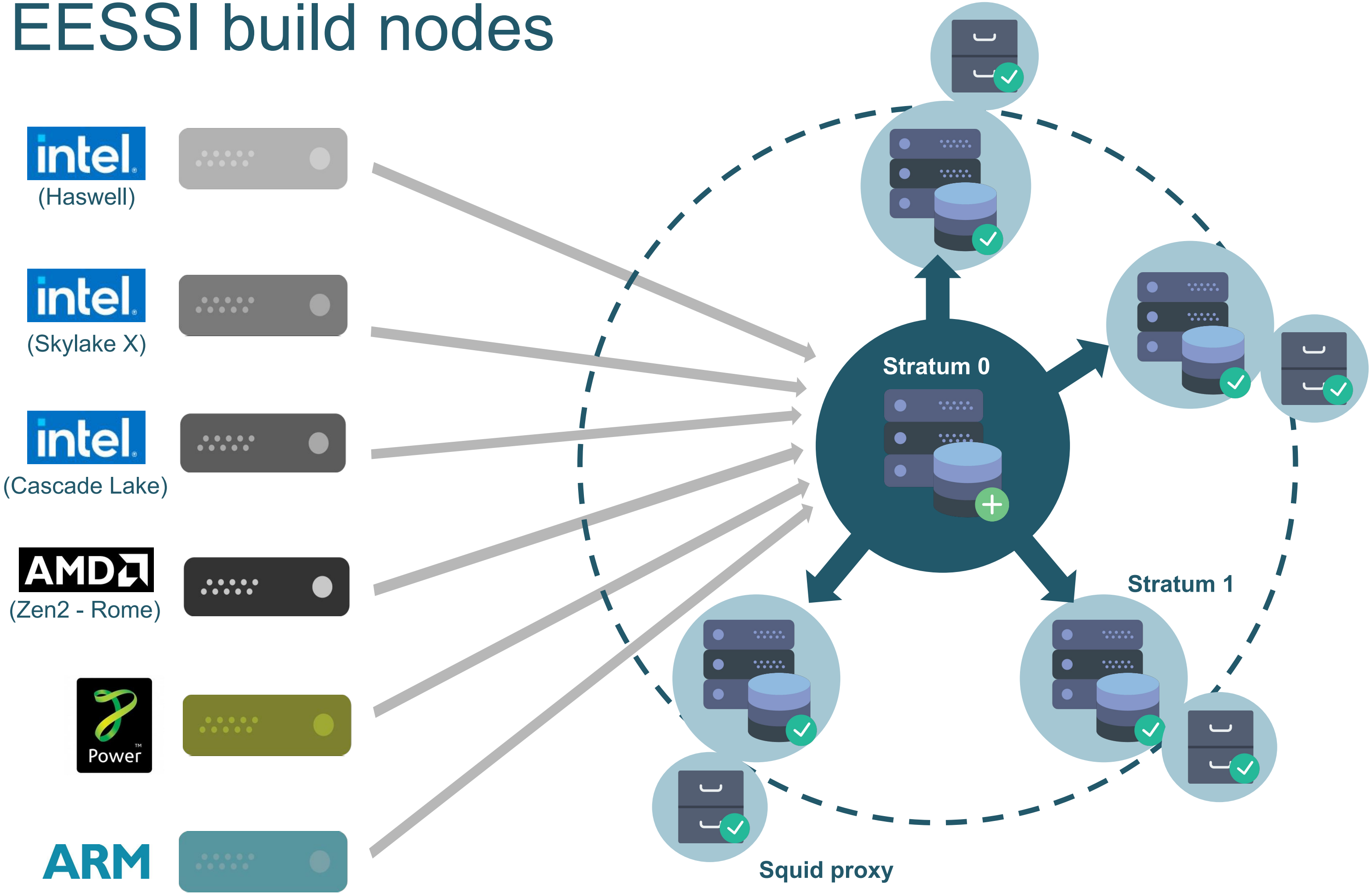


(see https://eessi.github.io/docs/pilot)

EESSI build nodes

- Software **must** be optimized for specific system architectures (CPU, GPU)
- Easiest way is to use different build nodes for this
  - Other options (cross-compiling, using QEMU, …) are more cumbersome/slower
- Software must be installed in `/cvmfs/...` (no relocation)
- Build nodes will most likely be hosted at different sites (important w.r.t. trust/security)
- Ideally:
  - Easy to set up build nodes “on demand” (containers!)
  - Isolation from host OS (containers!)
  - No root privileges required for mounting `/cvmfs` or (illusion of) write access
  - Building/installing software is done before starting a transaction to publish software

EESSI build nodes



EESSI
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

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

Publishing software into EESSI repository

- Publishing should be separate from building/installing software
    - Some installations take a (very) long time…
    - Installations may fail and may need to be restarted/fixed
- Publishing should be restricted to certain users / build nodes
    - Gateway - publisher approach
    - Ideally this can be done from the build nodes (without root privileges...)
- Future plan/hope: fully automated setup
    - Pull requests to EESSI GitHub repository to add software
    - Software installation + publishing is done automatically after review/approval
    - Powered by GitHub Actions? Build nodes produce RPMs to install to `/cvmfs`?
    - Experiences / suggestions?

Adding software: current approach (2020.08 pilot)

EESSI
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

- Minimal Singularity container

  (Base OS + CernVM-FS client + `fuse-overlayfs`)

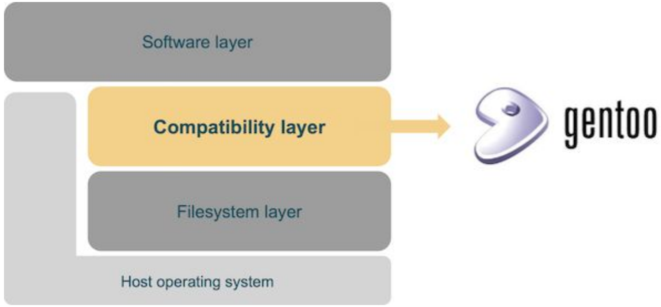
- Mount CernVM-FS repos using `singularity --fusemount`
  - Mount software repo at `/cvmfs_ro`, use as overlay's `lower` dir

- Use `fuse-overlayfs` with `--fusemount` to make a writable overlay
  - Merged mountpoint for software repo at `/cvmfs`
  - Build/install software into overlay (in container)
  - Create tarball of installed software from overlay's `upper` dir
  - Ingest the new software on a publisher node (Stratum 0 for now)

Questions related to build nodes

- Issue: only seems to work with ancient fuse-overlayfs versions
  - Weird "`Operation not permitted errors`" for `cd`, `mkdir`
  - https://github.com/containers/fuse-overlayfs/issues/232
  - https://groups.google.com/a/lbl.gov/g/singularity/c/2CobnkVUI0w
- Does `cvmfs enter` work in a similar way? Any suggestions?
  - Can/should we try using current CernVM-FS develop to play with this?
- More info on “`cvmfs push`” command (portal to directly push user payloads?)
- ETA for CernVM-FS 2.8? Do clients/server needs to upgrade in sync?
- How are Git branches managed vs future releases? (stable vs devel vs cvmfs-2.7)

Questions on build isolation



- Build environment should be fully isolated from host OS
- Only tools/libraries from EESSI compatibility layer should be leveraged
- Containers partially solve this: full control over OS + packages in build environment
- Also need to use compat layer as alternate “sysroot”

  (`--with-sysroot` in GCC, etc.)
- Is there experience in CernVM-FS community on this?

More questions

- Support for alien caches in CernVM-FS
  - Compute nodes with no internet access as clients
  - CernVM-FS only in Singularity container, not on host
  - Alien cache populated in container to fuse mounted GPFS dir
  - Works fine in serial, problems when **using MPI** (see issue):

    “`Failed to initialize loader socket`”
- Will Azure blob support be included in next CernVM-FS release?
  - https://github.com/cvmfs/cvmfs/pull/2590