Improving reproducibility of scientific software using Nix/NixOS

A case study on preCICE adapters and solvers

Max Hausch and Simon Hauser, 2024/03/06

unsplash

Max Hausch

- Master Student in final semester(s)
- Working at Helsinki Systems
- Uses Nix/NixOS as daily driver
- Open Source contributor

cheriimoya
st175425@stud.uni-stuttgart.de

Simon Hauser

- Master Student in final semester
- Part-time Software Developer
- Open Source Maintainer
- Nix/NixOS enthusiast

conni2461
st148883@stud.uni-stuttgart.de

This is John

- John is 26 Years old and just finished his Master Thesis
- John now wants to write his PhD Thesis
- He found an open position based on preCICE and got it 🎉

preCICE

- Let’s accompany him on his journey →

All persons mentioned in this story are fictitious

DAY 1 - Getting started

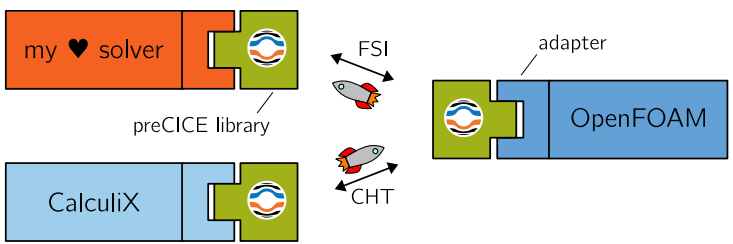
- John does some research about his topic
- Starts with the preCICE overview
- John is using Ubuntu
- Can install large parts of the software with `apt`
- Downloads `.deb` file from preCICE website
- Installation of preCICE successful 🎉

DAY 2 - Trying to install the rest

- A lot of copying instructions in the quickstart guide
  - `wget`
  - `sudo apt install`
  - `git clone ... && cd ... && ./Allwmake`
- Frustration and not knowing what is cloned and installed, why and how
- But got the quickstart example working 🥲
  - Still, there is a lot more that need to be built

DAY 3 - Why do we need to build so much

- John reads upon preCICE to understand what preCICE is
- preCICE is a coupling library for multi-physics simulations
  - Library that connects different solvers
- preCICE adapters need to be built on top of, or for solvers
- Many solvers need to be built beforehand



DAY 4 - preCICE distribution and VM

- The docs mention a "distribution" 🤔
  - Collection of adapter/solver versions that work together
- And there is a VM repo 👀
  - Its a Vagrant VM with provision scripts
- John tries out the VM
  - The VM provisions a really long time on his laptop
  - Still, everything seems to work
- New Strategy → rebuild most of VM locally so he can do local development
  - Using VM bootstrapping scripts

DAY 5 till 9 - distribution but locally without VM

- Locally building DUNE and OpenFOAM
- Needs to get familiar with both softwares
  - Custom build systems, no well-established ones like e.g. cmake
  - Easy to make mistakes e.g. end up with DEBUG build
  - Dune needs to be built in `` `$HOME` ``
- In the end he managed to get it going 🥳

DAY 10 till 27 - Researching and Working

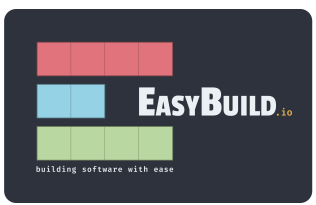
- John requests access to the HPC cluster
- He is researching papers and generally working on the project

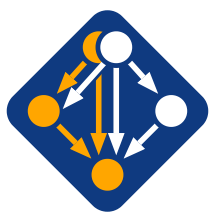
DAY 28 - HPC

- After a long time of institute bureaucracy John finally got his HPC access
- One ssh key setup later John tries to setup preCICE on the HPC
- Opening the docs and the VM scripts
  - `` `sudo apt install` `` fails
  - `` `ac123456 is not in the sudoers file. This incident will be reported.` ``
- And what now? 🤷

DAY 28 - HPC

- Let’s try to contact HPC admin first
  - Afterwards he goes for lunch and talks to some colleges
- Suggestions for EasyBuild or Spack to install software without root
  - Install instructions recommend Spack, but EasyBuild sounds simple and, well, easy
- Let’s try both and see what works best





DAY 29 - EasyBuild

- You can use it without root, which makes John happy ❤
- Docs are okay-ish, yet confusing
  - There are multiple different ways to install 😕
  - John simply tries the first option and it works; Easy!

DAY 29 - EasyBuild

- One EasyBuild install later
- `error: One or more OS dependencies were not found: [('openssl-devel', 'libssl-dev', 'libopenssl-devel')]`
  - Searches through the repo and finds `osdependencies`
  - That's kind of a problem without root 😢
  - Just in case, he writes another mail to HPC admin, first mail didn't get a response

DAY 29 - EasyBuild

- Installs a different package just to see if EasyBuild even works
  - Found another roadblock 😭
  - Requires setting runtime varibales
  - John doesn’t know why and it did cost a lot of time to find that solution
- Out of frustration, John gives up on EasyBuild 😡

DAY 30 - Spack - Please just work

- HPC admins answer is here and boils down to
  - It’s discouraged and he should look into different solutions
- So John looks into Spack and follows the install guide
  - It’s weird that the installed Spack version is simply the latest `develop` commit
- He installs preCICE
  - Spack interface feels similar to apt
  - But builds all dependencies from source
- He runs `binprecice` - seems to work out of the box 🚀

DAY 30 - Spack - Please just work

- He is interested in how Spack links so lets run `ldd` on that binary
- Links against
  - system libraries
  - libraries built using Spack

```
user@wolfgang-login:~$ ldd $SPACK_ROOT/gcc-12.2.0/precice-2.5.0-v5smeytzhaef6raifclgrpy7rh67tahy/bin/binprecice
        linux-vdso.so.1
        libprecice.so.2 => $SPACK_ROOT/gcc-12.2.0/precice-2.5.0-v5smeytzhaef6raifclgrpy7rh67tahy/lib/libprecice.so.2
        libstdc++.so.6 => $SPACK_ROOT/gcc-9.4.0/gcc-12.2.0-6dzyfotesxykmkszgjbdfppbrmdjktkx/lib64/libstdc++.so.6
        libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0×00007f555e80c000)
        libgcc_s.so.1 => $SPACK_ROOT/gcc-9.4.0/gcc-12.2.0-6dzyfotesxykmkszgjbdfppbrmdjktkx/lib64/libgcc_s.so.1
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
        libboost_log_setup.so.1.82.0 => $SPACK_ROOT/gcc-12.2.0/boost-1.82.0-fk4e5nbodijbtdsgk7z7ccx7yvsdztg3/lib/libboost_log_setup.so.1.82.0
        libboost_program_options.so.1.82.0 => $SPACK_ROOT/gcc-12.2.0/boost-1.82.0-fk4e5nbodijbtdsgk7z7ccx7yvsdztg3/lib/libboost_program_options.so.1
        libboost_system.so.1.82.0 => $SPACK_ROOT/gcc-12.2.0/boost-1.82.0-fk4e5nbodijbtdsgk7z7ccx7yvsdztg3/lib/libboost_system.so.1.82.0
        libboost_unit_test_framework.so.1.82.0 => $SPACK_ROOT/gcc-12.2.0/boost-1.82.0-fk4e5nbodijbtdsgk7z7ccx7yvsdztg3/lib/libboost_unit_test_framew
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2
        libxml2.so.2 => $SPACK_ROOT/gcc-12.2.0/libxml2-2.10.3-bkkigiua3njfqc4w5c47dnsn6njfp7i2/lib/libxml2.so.2
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0
        /lib64/ld-linux-x86-64.so.2
         ...
```

DAY 30 - Spack - Please just work

- John continues with the setup for the rest of the day
- Installing OpenFOAM using Spack
  - Installing adapter manually and linking against Spacks version of preCICE and OpenFOAM
- Sadly still has to build Dune manually, no Package available
- But ultimately somewhat happy he now finally has an HPC setup

DAY 31 - Further development locally

- John wants to work on his research locally, not remote via ssh
- Setting up Spack again
  - John interrupts compilation, as it takes a long time compiling ⌚
- Could continue to use `apt` versions of tools like `OpenFOAM`
  - not the same version as on HPC
- Looks at the preCICE website again and finds a section about installing with Nix
  - A friend already told him about Nix and NixOS
  - Promise of having the same software versions on different machines 🤔

DAY 32 - Using the Nix approach

- John tests the thing he read yesterday locally
  - With a single command, he can download a compiled `` `OpenFOAM` `` from a binary cache
  - All other official solvers and preCICE adapters are also there
- Without really believing it, runs the same command on HPC
  - It also works there!? 🔥

DAY 32,5 - Why does this work so well?

- John wants to know more about Nix, so he visits the website
  - Nix is a declarative package manager
  - Focus on reproducibility by hashing dependencies (similar to Spack)
  - Complete dependencies, version pinning
  - Self-contained, not depending on a "single" system library

DAY 32,5 - Why does this work so well?

- This makes him realize: EasyBuild and Spack builds are not reproducible ❗

```
[nix-shell:~]$ ldd /nix/store/hc69mg8l7f0fnhdvgyhpkh86wj9p0pd5-precice-2.5.0/bin/binprecice
        linux-vdso.so.1
        libprecice.so.2 ⇒ /nix/store/hc69m[ ... ]-precice-2.5.0/lib/libprecice.so.2
        libstdc++.so.6 ⇒ /nix/store/a3zlv[ ... ]-gcc-12.3.0-lib/lib/libstdc++.so.6
        libm.so.6 ⇒ /nix/store/1zy01[ ... ]-glibc-2.38-44/lib/libm.so.6
        libgcc_s.so.1 ⇒ /nix/store/a3zlv[ ... ]-gcc-12.3.0-lib/lib/libgcc_s.so.1
        libc.so.6 ⇒ /nix/store/1zy01[ ... ]-glibc-2.38-44/lib/libc.so.6
        libboost_log_setup.so.1.81.0 ⇒ /nix/store/fcb52[ ... ]-boost-1.81.0/lib/libboost_log_setup.so.1.81.0
        libboost_program_options.so.1.81.0 ⇒ /nix/store/fcb52[ ... ]-boost-1.81.0/lib/libboost_program_options.so.1.81.0
        libboost_system.so.1.81.0 ⇒ /nix/store/fcb52[ ... ]-boost-1.81.0/lib/libboost_system.so.1.81.0
        libboost_unit_test_framework.so.1.81.0 ⇒ /nix/store/fcb52[ ... ]-boost-1.81.0/lib/libboost_unit_test_framework.so.1.81.0
        libdl.so.2 ⇒ /nix/store/1zy01[ ... ]-glibc-2.38-44/lib/libdl.so.2
        libxml2.so.2 ⇒ /nix/store/vybhx[ ... ]-libxml2-2.11.7/lib/libxml2.so.2
        libpython3.11.so.1.0 ⇒ /nix/store/yvhws[ ... ]-python3-3.11.8/lib/libpython3.11.so.1.0
        libboost_log.so.1.81.0 ⇒ /nix/store/fcb52[ ... ]-boost-1.81.0/lib/libboost_log.so.1.81.0
        libboost_filesystem.so.1.81.0 ⇒ /nix/store/fcb52[ ... ]-boost-1.81.0/lib/libboost_filesystem.so.1.81.0
        libboost_thread.so.1.81.0 ⇒ /nix/store/fcb52[ ... ]-boost-1.81.0/lib/libboost_thread.so.1.81.0
        libboost_atomic.so.1.81.0 ⇒ /nix/store/fcb52[ ... ]-boost-1.81.0/lib/libboost_atomic.so.1.81.0
        libboost_chrono.so.1.81.0 ⇒ /nix/store/fcb52[ ... ]-boost-1.81.0/lib/libboost_chrono.so.1.81.0
        libboost_regex.so.1.81.0 ⇒ /nix/store/fcb52[ ... ]-boost-1.81.0/lib/libboost_regex.so.1.81.0
        libmpi_cxx.so.40 ⇒ /nix/store/c2g6p[ ... ]-openmpi-4.1.6/lib/libmpi_cxx.so.40
        libmpi.so.40 ⇒ /nix/store/c2g6p[ ... ]-openmpi-4.1.6/lib/libmpi.so.40
        /nix/store/1zy01[ ... ]-glibc-2.38-44/lib/ld-linux-x86-64.so.2 ⇒ /nix/store/1zy01[ ... ]-glibc-2.38-44/lib64/ld-linux-x86-64.so.2
```

DAY 32,5 - Deep dive

- To understand what happens, he takes a look at the precice/nix-packages repo
  - All official adapters and theirs solvers are packaged using Nix
  - Many problems using non-standard build systems are solved and documented there
  - Nix expressions representing the solvers, adapters and even a VM based on NixOS 😍
- Is amazed about Nix
- Through the case study in the repo on preCICE, he sees the value for Nix in research
  - Therefore he sticks to this approach throughout his work

YEAR 3 - Releasing his work

- John just finished writing his dissertation
- Now, he wants to make his work reusable
- As he has the whole Nix setup, he pushes everything to GitHub
- Referencing a single commit in his dissertation
  - Allowing for verification of results
  - Allowing to swap out only a single adapter (e.g. updating to another version)

John had a really good time after discovering Nix

This is Jack

- Jack just finished his Master Thesis and completed his SimTech master's degree
- Jack now wants to write his PhD Thesis
- He found a Thesis based on Johns dissertation and got it 🎉
- Let's find out how it goes →

DAY 1 - Gathering information

- Reading abstract and summary of Johns dissertation
  - Sees GitHub link
  - Reads `` `README.md` `` in the repo

DAY 2 - Setting up the environment

- Jack clones Johns repo
- Installs Nix
- Enters a Nix shell
  - This provides the same environment that John used
  - All software parts are pinned
- Can start right away, without the issues John experienced

Jack had a really good time after discovering Nix

TODAY - Making people interested in Nix

- This presentation is only scratching the surface
- If you are interested, read our README and our paper on GitHub
- Try out Nix without installing as root
  - Cleanup script removes everything cleanly
- Read up on Nix and take your first steps

The End

Thank you for listening

https://github.com/precice/nix-packages/