Considering pixi/conda for DUNE code

Jake Calcutt
NPPS Meeting
07/15/26

Introduction

DUNE has been in a transitionary period with its software build & package management system for a few years

- Previously used FNAL-maintained mrb and ups until SL7 EOL
- FNAL moving to AL9, decided not to maintain mrb/ups

Spack chosen as the replacement

- I have mixed feelings about spack, so I wanted to try out another option: pixi & conda
    - Particularly (IMO): Spack has a steep learning curve that would be a barrier for most users

Background

Saw a talk by Matthew Feickert at CHEP 2026 showing the usage of conda-forge to provide (HEP) software as conda packages

- Also demonstrated the use of tools like rattler-build to build conda packages and pixi for package management (pixi-build is also a preview feature)

I started to explore this as an option for DUNE

- Also used this as a chance to exercise using LLMs for assistance (I'd been slow to adopt)
    - IMO this was a *very successful* use-case for LLM tools

AI/ML Tooling

I used Claude w/ Opus 4.8 for a lot of this

- Its ability to do iterative dev/test near inexhaustibly was very helpful

I mostly wanted to get an end-to-end build going

- This meant some runtime issues persisted that I had not noticed until late in the game

It was able to solve a lot of finicky building issues but I did not do a good job documenting them

- Something to watch out for as I get used to using LLM tools

Will share some thoughts/experience on claude usage throughout 

Rattler-Build, Pixi, Conda-Forge

[rattler-build](#) 

Builds your software into a conda package

User writes a build.sh script and a recipe.yaml that steers the script, defines dependencies

[conda-forge](#) 

Repository for community-derived conda packages

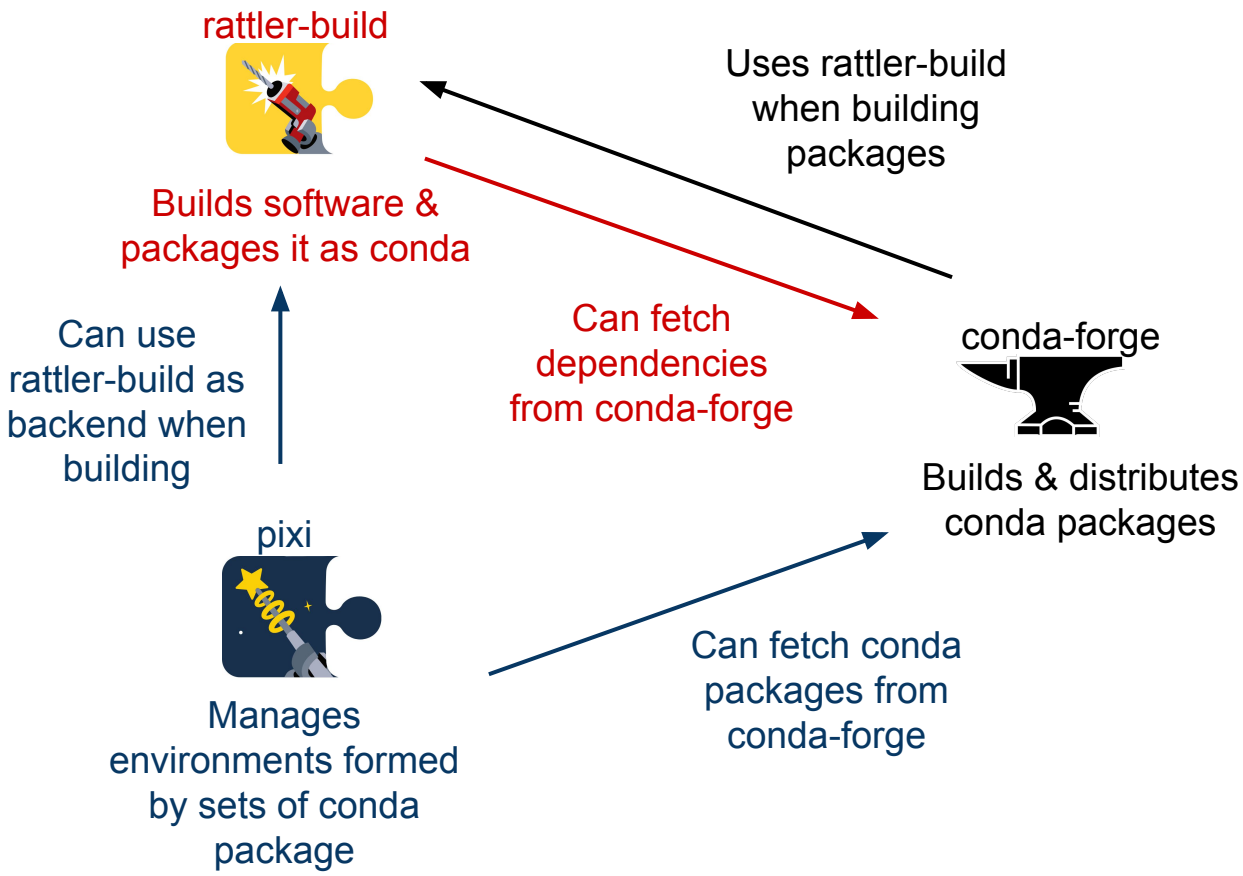
Automated builds, ABI migration

[pixi](#) 

Orchestrates environments made of conda packages

Can use rattler-build as a backend for building in preview features

How these fit together



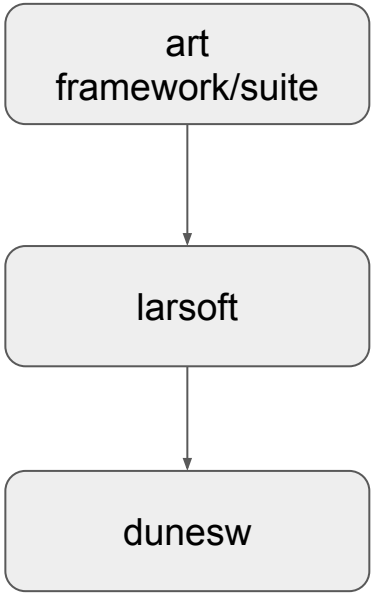
DUNE's Software Stack

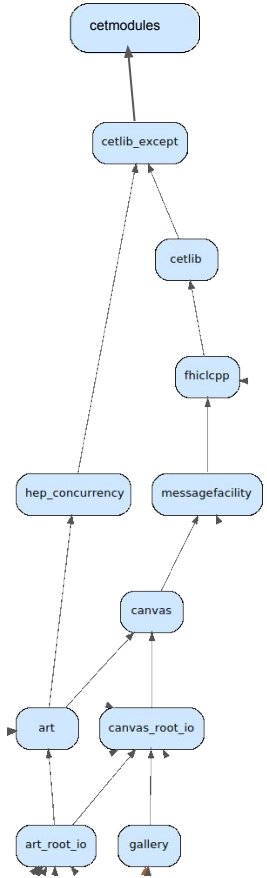
Planned to segment this into 3 broader steps:

1. Art event framework
   a. Mostly-lightweight packages c++ packages + CMake modules
2. Larsoft
3. dunesw

Made a plan with claude to work on these 3 parts separately

- Made furthermore plans for within each overall step





First steps

Target: put this all on conda-forge

- Requires rattler-build recipes
- pixi not necessarily a factor
- Needs to meet certain conda-forge specifications
- I’m starting to move away from this as a target per se
    - Rattler build recipes made, now want to design workflow for users to build/manage envs with pixi

Wanted to start small so I chose 2 small packages

1. Cetmodules
    a. FNAL-style CMake modules
    b. Install is a small cmake run to create modules
2. justin
    a. DUNE’s workflow system
    b. Install is just copying things to the install location

First steps

Target: put this all on conda-forge

- Requires rattler-build recipes
- pixi not necessarily a factor
- Needs to meet certain conda-forge specifications
- I'm starting to move away from this as a target per se
  - Rattler build recipes made, now want to design workflow for users to build/manage envs with pixi

Wanted to start small so I chose 2 small packages

1. Cetmodules
   a. FNAL-style CMake modules
   b. Install is a small cmake run to create modules
2. justin
   a. DUNE's workflow system
   b. Install is just copying things to the install location

Already on conda-forge!

- Another expt. (SHiP) using DUNE's new (in-dev) framework "Phlex"
- SHiP member put both on conda-forge already so I just needed to backport from a later version

Getting things onto conda-forge

1. Get package building with rattler-build
    a. Create recipe.yaml and build.sh (+possible extras needed by build)
2. Fork conda-forge/staged-recipes
3. Make a PR from a branch with package's build instructions
    a. Nice automation for linting to adhere to coding style specs + building/testing on different archs
    b. Iterate with conda-forge experts to determine if recipe is suitable for conda-forge
4. PR Merged → conda-forge "Feedstock" is created
    a. Instructions for conda-forge to produce a build of a package
        i. Includes variants + pinnings etc.

justin

Pointed claude to the existing spack recipe for justin

It handily replicated this in rattler-build style

- TBH: not hard, just a few copies + minor python & rucio-clients deps

Made PR to staged-recipes

- Iterated with conda-forge experts to brush up recipe
  - Get version number into SemVer etc.
  - Make name not so generic: dune-justin

Found an issue preventing me from running this on my Ubuntu laptop

- Patched in conda-forge builds + made PR to fix upstream

art

Again, pointed claude to existing spack recipes for basis

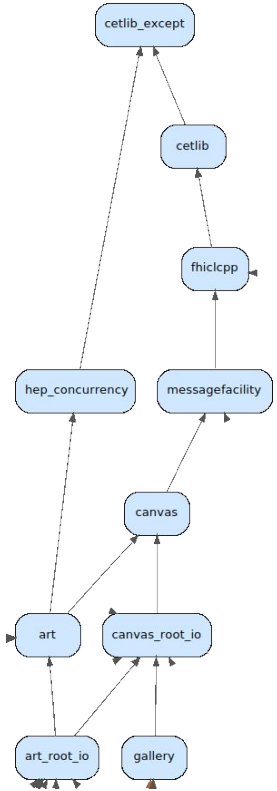
- Create recipe → build → next package

Hit some small barriers with conda-forge pinning newer versions of various deps

- Boost API changes
- libc++ transitively including various headers
- Drop -Werror compiler flag

Also small things like cetmodules installing INSTALL/README files into rattler-build install prefix

- Creates conflicts → add patch to prevent these being installed

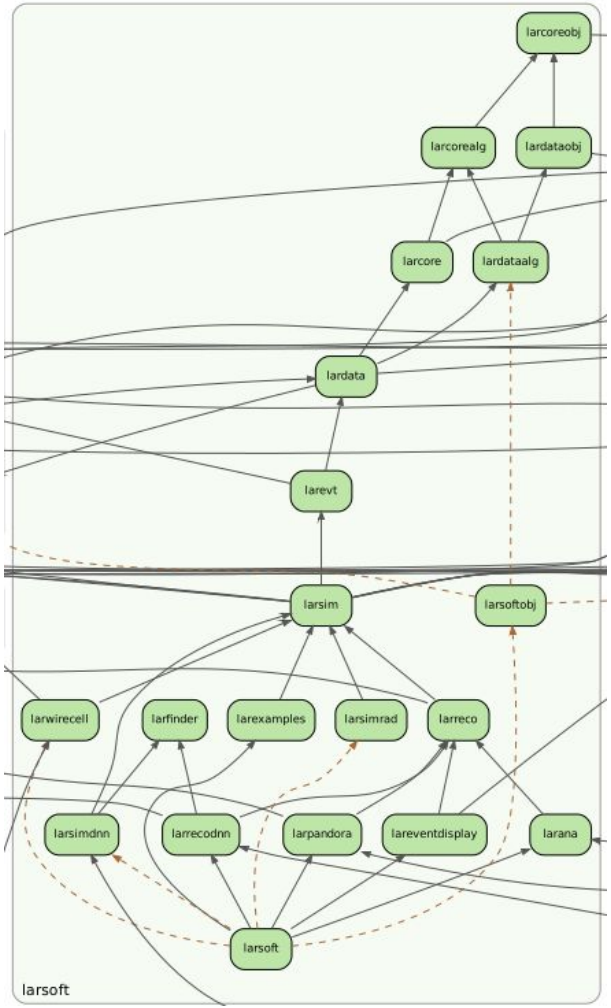


Larsoft suite

Larger set of packages needed to repackage

- 20 larsoft vs 10 art
- All are c++ packages

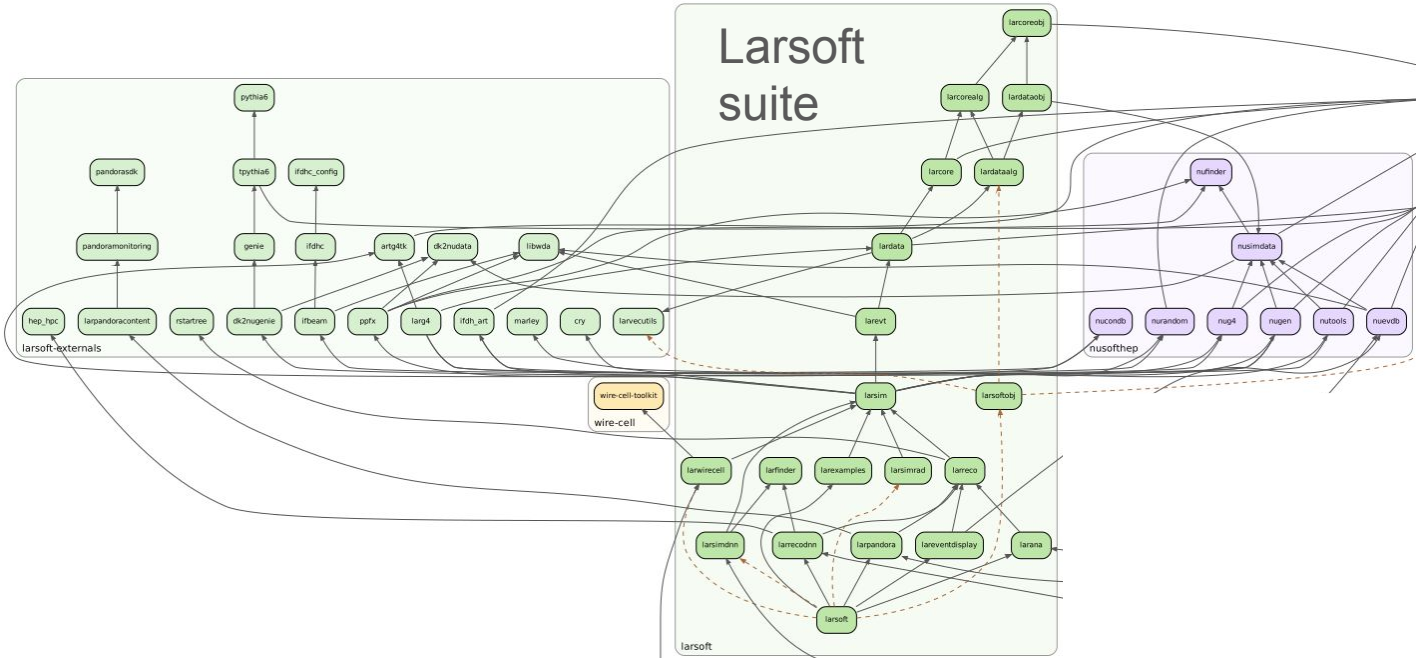
At first, a slightly larger effort but seemed not too bad



Larsoft suite cont.

But I forgot to consider all of the dependencies not on conda-forge…

- About equal to the number of actual in the larsoft suite itself



Larsoft Suite + Externals

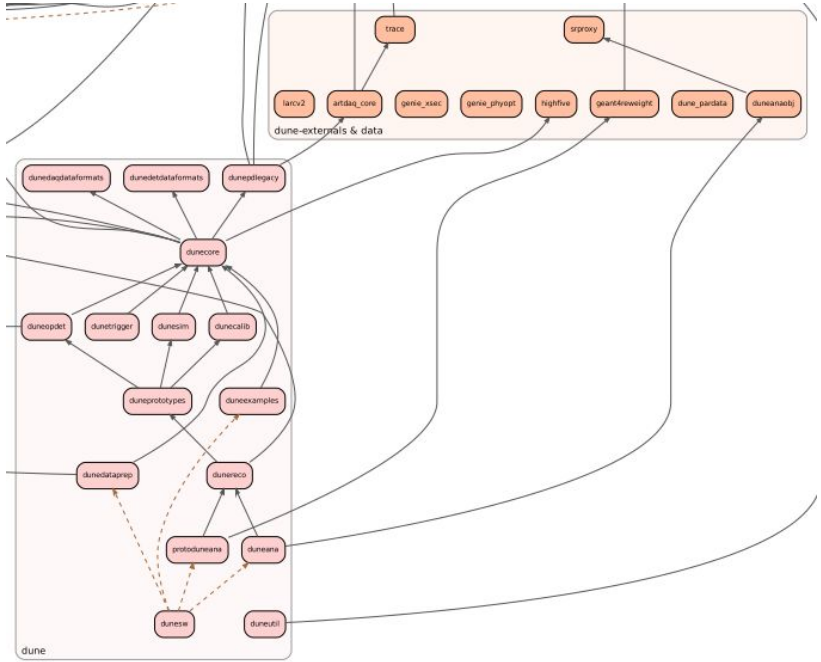
Some tricky bits/gotchas

- Conda-forge Geant4 pins boost < 1.91
  - Only came up midway through the suite, had to rebuild all previous parts (art + ~half larsoft)
- Genie (nu event generator) relies on TPythia6 shipped by root
  - Pythia6 dropped by root at conda-forge-pinned version
  - Claude figured out to package Pythia6 + ship its own TPythia6 interface
- Various ML package hangups
  - Tensorflow on conda-forge is too new for parts of larsoft
  - Torch_scatter not on conda-forge

DUNE Suite

Overall smaller set of packages

Still had some significant hurdles



DUNE Suite Gotchas

A dune suite ROOT dictionary includes a header from a larsoft dep (larcore), which includes a range3 header (uses ranges namespace)

- rootcling has “using namespace std” → c++20 includes std::ranges →namespace conflict…
  - Add a header guard in larcore to use std::ranges if c++20
    - Had to rebuild everything again…
    - But on the bright side: made a quick detour to submit a PR to larcore in order to fix this for future c++ std migration

DUNE Suite Gotchas

One DUNE package (dunereco) vendors some code from a neutrino-community ML project (DeepLearnPhysics/Supera)

- Against conda-forge guidelines, so needed to package supera

Some RooFit API changes + classes dropped

- For short term progress (getting the top-to-bottom stack built), dropped part of the stack that uses RooFit. Might be a good target to extract anyways, but in hindsight, I think the code that uses RooFit is deprecated/unused

Current Status

All of dune’s software stack builds along conda-forge pinnings

Some polishing remains

- Make sure runtime environment is good
  - Some path env vars are needed for configurations/auxiliary input files to be resolved
- Making decision about outdated parts of packages to extract/deprecate
  - Or get patched for conda-forge pinnings

Current Status cont.

Also need to consider: should we rely on conda-forge?

- Is it good policy to use pinned conda forge dependencies?
  - Can point to i.e. FNAL-distributed conda packages for dependencies to keep things more stable/controlled

I see pixi as very useful for most users + certain developers

- Next steps are to add a small “pixi manifest” to each recipe to get working with pixi-build → Design overall workflow for working in a development context

Overall Timeline

After some initial planning, exploration the overall port to ~90% done (need more polishing)