Spack and SpackDev for LArSoft

James Amundson

LArSoft Coordination Meeting

Nov 8, 2016

Outline

- Why?
  - Why mess with the build system?
- What?
  - What is Spack?
  - What is SpackDev?
- Which?
  - Which packages have been “ported to Spack?”
- When?
  - The $64K question.
    - That’s 65,536 USD!

Why?

Why mess with the build system?

Why mess with the build system?

Two big reasons

1. Recent releases of Mac OS have made it prohibitively difficult to rely on LD_LIBRARY_PATH.
   - UPS relies on “LD_LIBRARY_PATH”.
     - Really DYLD_LIBRARY_PATH on Mac OS.
     - We use UPS for *everything.*
2. The current build system is the source of many complaints.
   - People hate death, taxes and build systems.
   - Some are still better than others.

“LD_LIBRARY_PATH” and RPATH

- Environment variables are good for user preferences. They are fluid.
  - Good for things that are not known at compile time
- Compile A, linking against library B
  - When you run A, do you consider the location of library B a user preference? Wasn’t it known at compile time?
    - Plugins are a different story.
- “LD_LIBRARY_PATH” is not inherited in OSX El Capitan and later
  - Whether we like it or not
- With RPATH, the location of library B is stored in A
- What about plugins?
  - If calling dlopen, you can use your own search mechanism

Build system complaints

- Many usability issues
- Perceived as slow
  - Build system can be slow because it is building more than necessary
    - Including user “error”
      - HPI tells me that “error” is the wrong way to look at it
  - Build system can be slow because the user finds it too complicated to use his/her own system
    - Log in to slow shared machine instead

Software Tapestry

- Need to consider entire software “tapestry”
  - external products
  - internal products
  - versioning
  - packaging
  - runtime environment
- We use ups for *everything*
  - specifically:
    - user environment variables: PATH, LD_LIBRARY_PATH, others
    - external package discovery
    - dependency/version management
    - listing of available software
  - tightly coupled
    - *art* depends on UPS Boost, not just Boost
- The ups user interface is awful

What?

What is Spack?

What is SpackDev?

What is Spack?

- Spack is (was) the Supercomputing Package manager
  - Meets our supercomputing needs! – also Linux machines, OSX
  - Has a rigorous model that includes multiple platforms, compilers and "qualifiers" – Solves the hard problem first
  - Heavily uses rpath
    - We are supplying functionality to provide relocatability
  - We are supplying a binary distribution method
  - Environment management is factored out
    - Multiple options: modules, dotkit. lmod in development.
  - Has a community including multiple developers from multiple communities
    - HEP Software Foundation Packaging Group is working on Spack as a HEP-wide solution
  - Very polished user interface
  - Excellent documentation
  - Excellent code quality

https://github.com/LLNL/spack

Replace UPS with Spack?

- Is the plan to replace UPS with Spack?
  - NO!
- Create a more loosely coupled system.
- Keep UPS for environment management
  - For the time being, at least as an option
    - Spack currently fully support the “modules” system.
  - Spack needs to be extended to write UPS table files
  - Should greatly simplify transition
- Include multi-package build system, SpackDev
  - Similar in spirit to MRB
- Modify cetbuildtools
  - Much work initially done by Ben Morgan!
  - More work done by Patrick Gartung.

What is SpackDev?

- SpackDev does what Spack does not
  - Spack is designed to build and install
    - Incremental builds are not part of the model
  - While Spack does allow for hand builds of packages…
    - …hand builds are not necessarily identical to those performed by Spack
    - …no easy way to work on more than one package at once
- SpackDev is a development manager for packages installable with Spack
  - SpackDev *uses* Spack; it is not *part of* Spack
  - SpackDev assists in the compilation of one or more dependent packages
- The packages themselves depend on neither Spack nor SpackDev

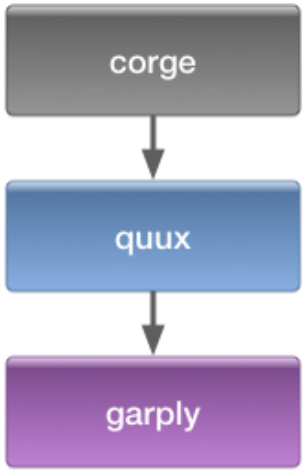
The plan for SpackDev is to provide a build system initially meeting the needs of *art* and LArSoft, but increasing this scope is a definite possibility.

SpackDev Creates a Build Environment

- Installs external dependencies
- Enables incremental builds of dependent packages
  - Dependencies handled correctly
- Creates a build area that depends only on standard build tools (CMake, Ninja and/or Make)
- Builds exactly as Spack would
  - including Spack's compiler wrappers
- IDE-friendly
  - IDE's such as QtCreator and CLion can be used for development

SpackDev Examples

- *spackdev init corge*
  - installs quux and garply, then creates build area for corge
- *spackdev init corge quux*
  - installs garply, then creates build area for corge and quux
- *spackdev init corge garply*
  - creates build area for corge, quux, and garply
    - SpackDev deduces that including quux is necessary for consistent builds, so it automatically adds it to the build area



Fermilab

Which?

Which packages have been ported to Spack?

Which LArSoft Packages Have Been Ported?

- If you are paying attention, this question may confuse you.
  - Packages should not depend on Spack at all.
  - “Porting” really means writing a Spack package description.
    - Generally very simple.
    - Spack itself contains package descriptions for over 1000 packages.
    - HSF packaging people have already ported an additional set of core HEP packages (most of which have become part of Spack.)
  - In our case, “porting” also means rearranging the build system to not depend explicitly on UPS.
    - This is the bulk of the work.
- The short answer is larcoreobj, including all packages which it depends upon, including *art.*

When?

When can you use SpackDev for LArSoft work?

When?

- SpackDev is not yet available to anyone.
- We are shooting for early user availability in January 2017.
  - Early users are expected to provide feedback.
  - A few earlier adopters may be invited.
- The largest difficulties arise in integrating with release and distribution.
- Spack/SpackDev availability for production use will depend on our experiences with early users and early distribution.

Questions?

- Feedback from end users is strongly encouraged!