Status of Spack Development / Migration

LArSoft Coordination Meeting 2018-08-14

Chris Green, FNAL.

Then …

- Previous status report (2018-05-22): https://indico.fnal.gov/event/17180/contribution/1/material/slides/1.pdf
- Quick recap:
  - Looking for a long term replacement to the elderly UPS-based ecosystem with wide applicability for HEP experiments.
  - Spack (recent development from the HPC realm) looks like a viable tool upon which to base a package management, build and release system.
  - Also need a system to facilitate multi-package development and user-level analysis operations (*a la* SoftRelTools, MRB, *etc.*): SpackDev.
  - Putting together a Minimum Viable Product to provide a first look.
  - External dependencies satisfied, preliminary “UPS-free” CMake-based build system (`cetmodules`), working up the art suite to produce standalone builds of each package with Spack.

… and now

- Since then:
  - Successful build of all art suite packages as individual recipes with Spack.
  - Overhaul of `cetmodules` to enhance facilities for configuration of non-compiled installation products (data, FHiCL, GDML, Perl files, *etc.*) and automatic location thereof by dependent packages.
  - Significant improvements to SpackDev to account for more demanding requirements of art suite and other packages versus the ones against which it was developed:
    - Non-default package configurations (both development-level and dependencies): versions, variants, *etc.*
    - Non-default development packages builds (CMake arguments, *etc.*)
    - Environments and testing.
    - VC systems other than git for package checkouts.

An excursion: the "SpackDev way” vs the “MRB (or SRT) way”

- “Tired:” multiple packages are combined for an integrated build.
  - If managed correctly, maximum parallelism can be achieved.
  - Natural development / analysis-level environment.
  - Strong requirements on coupling between individual package build systems and the overall build system: “special knowledge,” ongoing requirements on package developers, enforced common environment.
  - Some classes of problem can go unnoticed until after release.
- “Wired:” packages are developed, “together but separately.”
  - Packages can use their own build systems without coupling to the wider system.
  - More classes of problem are caught at development time.
  - Developer works in one package development environment at a time, executing global builds less frequently.
  - “Global build” includes installation of each package for use by dependents.

SpackDev: details

- One can make a SpackDev development area and populate it with packages to develop from any source that Spack understands.
- Additional packages required for a consistent development build are identified and downloaded.
- Dependencies are identified and built to the required configuration as necessary.
- A top-level `CMakeLists.txt` is generated, treating each package to be developed as an external package with its own build system.
- Parallel global build: intra-package parallelism, and inter-package parallelism at package level (subject to inter-package dependencies).

SpackDev: caveats and work remaining for MVP

- Packages to be developed must currently utilize CMake (fixable post-MVP).
- The global build cannot execute tests due to environment requirements (limitation).
- New command required: `spackdev env <package> [<cmd>]` to set up the required environment for a particular package under development, or execute a command within it.
- Minor enhancement to dependency build management necessary to improve robustness in some cases.
- MVP instructions and setup scripts are required.
- Barring surprises, the MVP should be available by the end of the week. If you need external packages over and above those required by the art suite itself in order to have a meaningful test of the system with your code, let me know (Geant4, Pythia, *etc., etc.*) N.B.: Nutools / LArSoft was not an anticipated part of the MVP.