Updates on Spack

James Amundson and Patrick Gartung

HSF Technical Forum - Packaging Discussion #10

2 Nov 2016

Current and Upcoming Spack Releases

- v0.9.1 latest tag from May 2016
  - v0.8.6 tag declared latest release
  - https://github.com/LLNL/spack/releases
- v 0.10 (1.0) ready for SC2016
  - https://github.com/LLNL/spack/projects/1
  - presumably will be merged from develop to master???
  - *(SC is even more important in the supercomputing community than CHEP is in computational HEP)*

HEP Contributions Accepted

- Brett Viren's view branch was accepted
- Patrick Gartung's branches
  - root6, clhep, xerces-c, geant4 package definitions accepted.
  - Minor changes for gcc build accepted
- Benedikt Hegner's branches
  - Package additions accepted.
  - config.yaml additions accepted.

HEP Contributions Still in Queue

- https://github.com/LLNL/spack/pull/1671
  - Patch to gcc build to add @rpath and make it relocatable
- https://github.com/LLNL/spack/pull/1013
  - Refinement of “Binary cache” tarballs for macOS
- https://github.com/LLNL/spack/pull/445
  - “Binary cache” tarballs
  - This patch is under review by security experts
    - Supercomputer people are very touchy about security
- https://github.com/LLNL/spack/pull/943
  - LD_LIBRARY_PATH for compilers
- Promise to integrate “real soon now”.

PG's Maintenance Experience

- There are too many changes to primary files to easily track on the feature branch. Every merge from develop can cause a breakage.
- There is no central mirror for source tarballs. The source version gets updated and older versions are not available on the original site. I have to update package dependency versions with each new rebuild.
- Other contributors are constantly updating versions of packages HEP software depends on. I have to maintain my own copy in the hep-spack repo.

Things to consider for moving forward with Spack

- We should consider maintaining an official HEP-SF branch
  - Periodic merge with official
  - Hard to imagine relying on official release schedule
    - Our constraints are very different than those of some other stakeholders
    - Experiments typically do this already
  - Goal would be to avoid divergence
    - *Not to maintain separate functionality*
- We should also consider which things go *into* Spack and which things work *with* Spack
  - Perhaps binary distributions should be external?
    - Just a question at this point

SpackDev

- SpackDev is a development project at Fermilab
- SpackDev is a development manager for packages installable with Spack
  - SpackDev uses Spack
    - i.e., the *with* model, not the *into* model
  - 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 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 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