Build system explorations: Spack

Patrick Gartung

Art Users Meeting

17 June 2016

Why look into new build systems?

- The current build system used to build Fermilab projects
  - Makes use of cmake macros that were developed with earlier versions cmake that lacked the features of current cmake
  - Makes use of environment variables defined by UPS, a Fermilab developed environment setup tool, that has problems running on new OS's and in linux containers
  - Only got a unified build script within the last year or so.
  - Doesn't integrate well with IDE's available on Linux or OSX
  - Uses LD_LIBRARY_PATH to find libraries
    - Great for portability
    - A problem on OSX ElCapitan where DYLD_LIBRARY_PATH is squashed in shell sub-processes by SIP
  - Not used anywhere else in HEP (but what build system is?)

What is Spack?

- Spack is the “supercomputer package manager”
  - Developed at LLNL for use on “supercomputers”
  - but it can be used on any Linux and OSX
- Spack was presented at SC15 and caught the attention of Fermilab developers
- Spack is like cmsbuild, buildFW, contractor, worch, lcgcmake, conda, macports, homebrew, etc in that it
  - Builds a stack of dependent software packages
- Spack is not like scram, mrb, setup_for_development, etc in that
  - Spack does not set up an environment for interactively building software (as of v0.9.1)
    - (That is where SpackDev comes in)

What is Spack?

- Open source, well documented and community supported
  - https://github.com/LLNL/spack source code
  - http://software.llnl.gov/spack documentation
  - https://www.computer.org/csdl/proceedings/sc/2015/3723/00/2807623.pdf official paper
  - https://github.com/LLNL/spack/wiki Information including info about weekly teleconference
  - https://groups.google.com/d/forum/spack google group

Spack Packages

- Each software package is defined by its own python class
  - Source versions and urls
  - Variants
    - Used to control cmake or configure options
  - Package dependencies
    - syntax for a range of dependency versions and variants
  - Source patches
  - Build and install method
- Spack packages can be collected into a repo that is added on to spack rather than adding to the 400+ spack built-in packages.
  - One was created by HSF to collect packages common to HEP
  - https://github.com/HEP-SF/hep-spack

Spack package build environment

- Spack sets up a build environment per package
  - explicitly unsets LD_LIBRARY_PATH in the package build environment
  - compilers set to wrapper scripts that
    - Add rpath to compiler and linker arguments for each package dependency
    - Add include paths to compiler arguments for each package dependency
  - sets CMAKE_PREFIX_PATH and/or PKCONFIG_PATH for each package dependency
- Why use rpath?
  - setting rpath circumvents the issue of SIP on OS X 10.11 squashing DYLD_LIBRARY_PATH in shell subprocesses
  - your program always finds the right libraries regardless of environment variables

Feedback and Experience

- Spack worked out of the box on SLF6, SL7 and Ubuntu 14.04
  - Spack includes the python packages it needs so the only requirement is python 2.6+
- Digging a little deeper than README.md resolved any initial misunderstandings.
- Spack worked out of the box on OS X with Xcode command line tools installed (clang)
  - Fortran support on OS X requires a Homebrew or MacPorts install of gcc with gfortran, but that's expected
- Spack developers and community are very helpful.
  - Google group and weekly teleconference (report from HEP community included in the agenda)
- Resolved a bug I found compiling gcc on OS X.
  - Spack compiler wrapper generated command "ld -r -rpath …".
  - On linux the -rpath is ignored, on OS X this errors out.

Platforms built on

- Using the root and geant4 packages definition in hep-spack, built on these platforms
  - OSX10.10 with clang 7.0.2 and spack-built gcc 4.9.3
  - OSX10.11 with clang 7.0.2
  - SL7 with spack-built gcc 4.9.3
  - SLF6 with spack-built gcc 4.9.3
  - Ubuntu 14.04 with spack-built gcc 4.9.3

Adding features to spack

- Adding features is straightforward and the spack developers accept many pull requests
  - Create tarballs and relocate pre-built binaries
    - Work by Benedikt Henger with testing by Patrick Gartung
      - https://github.com/LLNL/spack/pull/445
    - Refining package relocation on OSX by Patrick Gartung
      - https://github.com/LLNL/spack/pull/1013
  - Alternate install location
    - Work by Benedikt Hegner with testing by Patrick Gartung
      - https://github.com/LLNL/spack/pull/908
  - Create view directories ala lcgcmake
    - https://github.com/LLNL/spack/pull/869

Incompatible dependencies

- Question during HSF workshop: Does spack catch incompatible dependencies (i.e. different version requirements)
- Made root dependent on one version of clhep and geant4
  - I know root is not dependent on geant4. This was a test.
- Made geant4 dependent on a different version of clhep.
- Did spack catch this?
- Yes, but the error is a little cryptic:

```
[vagrant@localhost geant4]$ spack install root
==> Error: Invalid spec: 'clhep@2.3.2.2^cmake@3.2:'. Package clhep requires version 2.3.1.1, but spec asked for 2.3.2.2
```

Can Spack build Art?

- Sort of, but not with existing cmake scripts

- Needed some modification/replacement of cmake scripts
  - There are no UPS defined environment variables in the spack build environment
    - These could be defined by declaring the spack-built packages to a ups install, but I don't know how…

Can Spack build the Art stack?

- Ben Morgan has put in a lot of effort into building the art stack with Spack
  - Defined a Spack package repo with packages for cetlib, fhicl-cpp, messagefacility and canvas
    - https://github.com/drbenmorgan/artstack-spack
  - All dependent on cetbuildtools2
    - Attempt to redefine cetbuildtool macros without using environment variables defined by UPS
      - https://github.com/drbenmorgan/cetbuildtools2
      - http://drbenmorgan.github.io/cetbuildtools2/
  - Cetbuildtools macro find_ups_package() can be replaced by cmake's find_package() because Spack defines CMAKE_PREFIX_PATH and/or PKGCONFIG_PATH
- Out of the box I used this to build cetlib,fhicl-cpp and messagefacility

Can Spack build canvas/art?

- I picked up where Ben left off with canvas
- I reused the altCmakelists.cmake files I had created for the worch build of Art 1.14 and updated them for the latest canvas and art.
- I added the missing cmake cetbuildtools macro Build_Dictionary
  - Initially this crashed during dictionary generation because of a missing argument to the genreflex –D option
  - This missing argument was defined by cetbuildtools using ups environment variables
  - Needed to set one environment variable in the root package definition to the checkClassVersion script would work on linux.
- Built all of the libraries for canvas and art on SLF6 and SLF7.

Can Spack build Art on OSX?

- With gcc no. With clang mostly.
  - Spack wants to build everything with gcc, but the build of cmake with gcc fails. I am still trying to find a solution.
- Ben was able to build everything up to canvas with clang.
- I was able to build all of the libraries for canvas and art with clang and the changes I made for Linux.
- However, there is a problem loading libraries with root. The dependent libraries cannot be located even though the rpath to them is defined.

SpackDev

- Need to set up an interactive development environment. This is where SpackDev comes in.
- SpackDev
  - Thin layer atop Spack for building sets of dependent packages
  - Locates dependencies
  - Builds all dependent packages
    - If A->B->C and A->C and I modify C, A and B will be rebuilt
  - Transparent – all SpackDev commands written to file
  - SpackDev command files can be reviewed and executed separately
  - There to help you but not required. You can run the cmake commands yourself in the build environment spack sets up.
  - Build Cmake packages and non-Cmake packages
  - Cmake with or without (modified) cetbuildtools

Extra Slides

gSystem->Load error on OSX

root [5] gSystem->Load("libcanvas_Persistency_Common.dylib")

cling::DynamicLibraryManager::loadLibrary(): dlopen(/Users/gartung/spack/opt/spack/darwin-x86_64/clang-7.0.2-apple/canvas-dev-aahvlxyt73rnfjo5l2pv3pz7ldgcysgl/lib/libcanvas_Persistency_Common.1.03.01.dylib, 9): Library not loaded: libCLHEP-2.3.2.2.dylib

Referenced from: /Users/gartung/spack/opt/spack/darwin-x86_64/clang-7.0.2-apple/canvas-dev-aahvlxyt73rnfjo5l2pv3pz7ldgcysgl/lib/libcanvas_Persistency_Common.1.03.01.dylib

Reason: image not found

(int) -1

root [6] gSystem->Load("libcanvas_Persistency_Common_dict.dylib")

cling::DynamicLibraryManager::loadLibrary(): dlopen(/Users/gartung/spack/opt/spack/darwin-x86_64/clang-7.0.2-apple/canvas-dev-aahvlxyt73rnfjo5l2pv3pz7ldgcysgl/lib/libcanvas_Persistency_Common_dict.dylib, 9): Library not loaded: libCLHEP-2.3.2.2.dylib

Referenced from: /Users/gartung/spack/opt/spack/darwin-x86_64/clang-7.0.2-apple/canvas-dev-aahvlxyt73rnfjo5l2pv3pz7ldgcysgl/lib/libcanvas_Persistency_Common_dict.dylib

Reason: image not found

(int) -1

Can Spack be used to build LArSoft?

- I haven't had time to try that yet.
- In principal it should be straightforward to build the larsoftobj packages nusimdata, larcoreobj, lardataobj and larsimobj
- The rest of larsoft can be built once I can build the packages that the rest of larsoft depends on.