Modern Software Stack Building for HEP

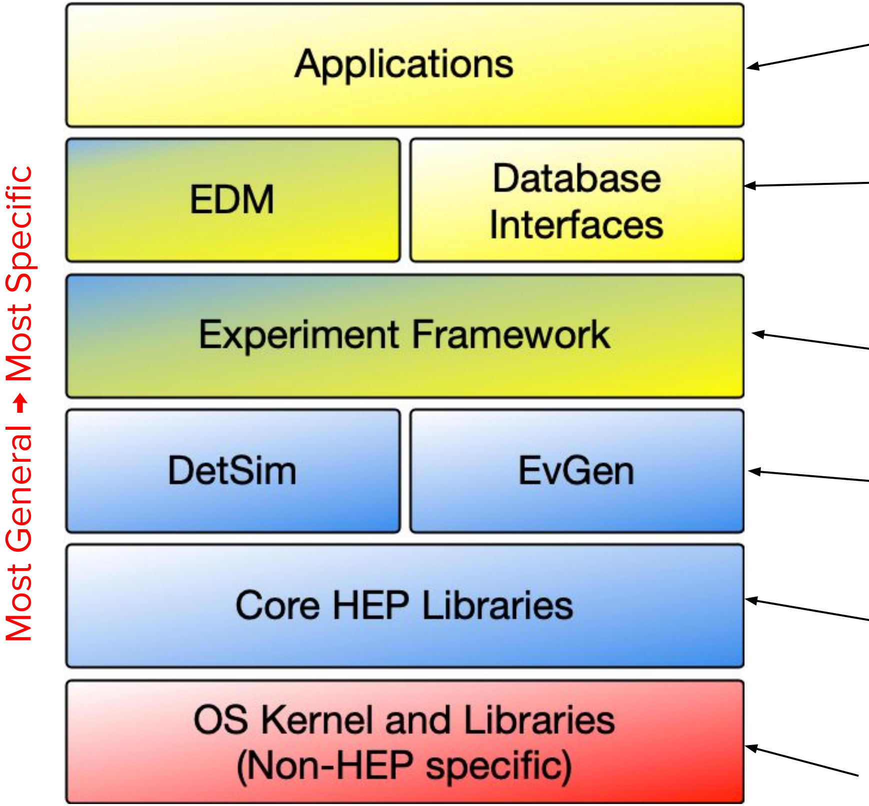
Graeme A Stewart, Ben Morgan, Javier Cervantes Villanueva, Hobbs Willett

2019-11-05

Software Building and Packaging for HEP

- Software is one of the central pillars of HEP experiments
- We have a wide range of requirements on our software, covering diverse use cases
  - Event generation, Simulation, DAQ, Reconstruction, Analysis
- HEP software lives as a connected series of packages
  - `tar -x … foo && make && make install` just won't do
  - In other words *no package is an island*
    - Dependencies on already installed pieces of software, often coming from the underlying distribution as well as other built dependencies
  - These dependencies have to be found by the build system of any package
  - A most sophisticated *build orchestrator* will check for these dependencies and pre-build them on demand
- This consistent set of packages, built in harmony, we refer to as a *software stack*

HEP Application Software



Application layer of modules/algorithms/processors that perform physics tasks (some generic examples like FastJet, Acts and PandoraPFA)

Usually experiment specific libraries for data representation and access: e.g. xAOD, LCIO; also detector specific conditions data

Experiment core orchestration layer, where everything else plugs in: Gaudi, CMSSW, Marlin

Specific components used by many experiments: Geant4, DELPHES, Pythia, ...

Provide core functionality widely used: ROOT, HepMC, HepPDT, DD4hep, ...

Many widely used non-HEP libraries: Boost, Python, Zlib, CMake, ...

HEP Software Stacks and Deployment

- HEP software stacks, in common with many software projects, need to maintain multiple versions
  - These versions generally evolve their external dependencies as well
  - Unlike other projects these versions usually have to be maintained for many years
- Build system must be able to support and patch stack versions years after their original deployment
  - External dependency issues can occasionally be the issue requiring patching
  - Significant trouble can arise when an underlying OS distribution dependency goes out of support
- Deployment is a closely coupled problem to the actual build
  - Our lives have been hugely eased by the widespread adoption of CVMFS and container technology

HEP Software Foundation Packaging Working Group

HEP Software Foundation

[link]

- Packaging and deploying a software stack is a problem faced right across HEP and the wider scientific community
  - Every experiment and software group has to put effort into doing this
  - Naively it seems an easy problem, but it quickly gets complicated and seemingly obvious solutions don't meet requirements
- Motivated formation of WG in 2015 as a forum for working together to improve
  - Knowledge sharing on tools and workflows in and outside HEP
  - See talk by Ben Morgan at CHEP 2018
- We looked at many tools - general FOSS, scientific community, HEP specific
  - We extracted use cases and provided bootstrap instructions to try out a number of tools
  - Focus now moved to implementation of stack using the most promising candidates...
    - Group continues to meet regularly for progress reports and to exchange information

Quick Summary of Desiderata...

- Support NxM complexity
  - Software versions
  - Architectures (and micro architectures), with build options
- Reproducibility
  - Capture all dependencies reliably
    - Minimise/eliminate dependency on underlying OS distribution
- Relocatable build products
  - Should not be tied to one install path at build time
  - CVMFS, container, local install, ...
  - Binary build products
- Runtime environment setup
  - Production and developer use cases differ slightly, both must be supported

Spack

spack.io

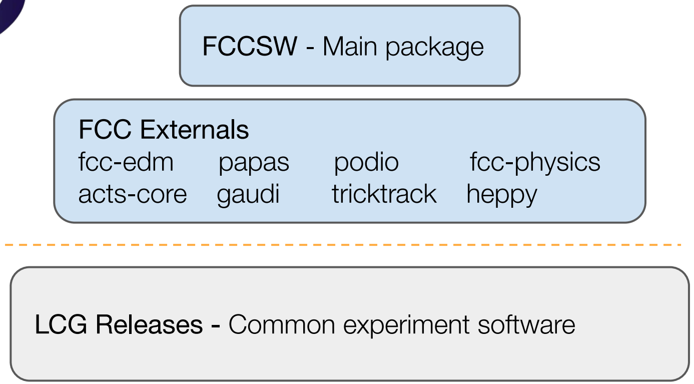
- Package manager and build orchestrator developed at Lawrence Livermore National Laboratory
- Originally developed for installing software to HPC systems
  - Strong emphasis on scientific software
- Supports multiple versions of software concurrently
  - Appends build hashes to install locations, RPATH used to resolve the correct dependencies
  - Common dependencies are shared
  - Support for different compiler toolchains as a core concept
- Dependencies are found and installed automatically
  - Full specification of all build options for dependencies supported
  - Will rebuild or install from existing binary build products
- Configuration on command line or from YAML files
  - Package descriptions written in Python
- Large community of contributors, supporting 3.5k packages
  - Active HEP sub-community (and Slack channel)

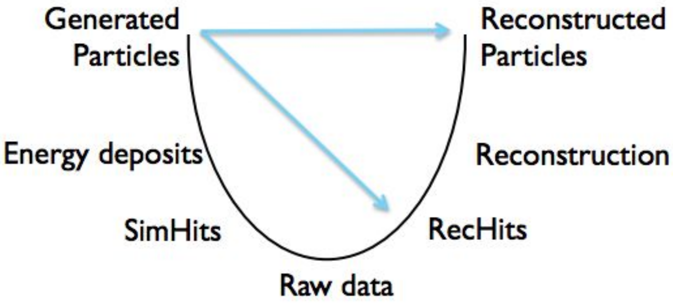
Future Circular Collider

- FCC project aims at a next generation collider in a circular tunnel of ~100km at CERN
- FCC software stack is not huge, but builds on top of an existing CERN built software stack
  - LCG Release
- Instructing Spack to take software pre-built in another build system is done:

```
root:
     buildable: false
     paths:{
            root@6.14.04%gcc@6.2.0
            arch=x86_64-centos7:
            /cvmfs/sft.cern.ch/lcg/releases/LCG_94/
             ROOT/6.14.04/x86_64-centos7-gcc62-opt
           }
```

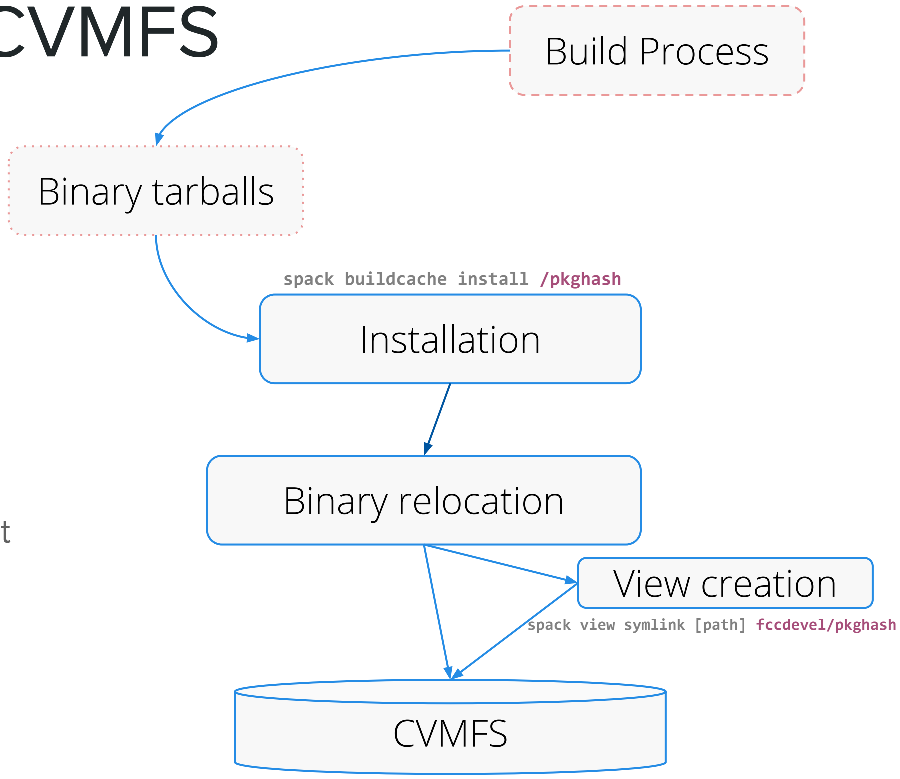
  - Same technique can be used to take packages from the OS (or anywhere else)





Build, Cache and Deploy to CVMFS

- Output of a build is a binary tarball
  - Put this in a cache visible to the CVMFS server
- On CVMFS server run Spack to install the buildcache binary
  - Buildcache was a HEP contribution to Spack
- Relocation is done at this stage
  - `patchelf` to update RPATHs
  - `sed`–esque process for configuration and other text files
- Issue: have to use the same platform as the target to ensure correctness
  - Docker containers were a workaround when target OS != CVMFS master OS
  - Enhancement now done



Spack for SuperNEMO

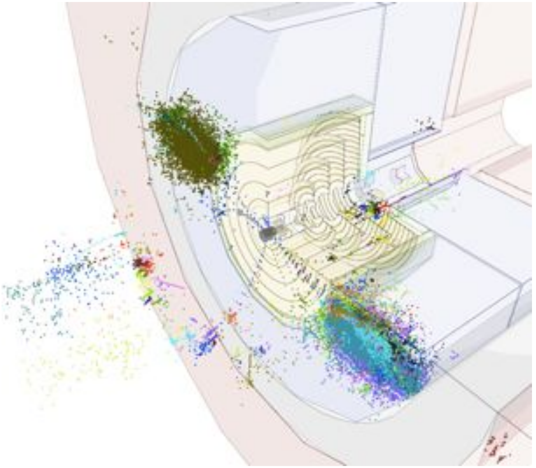
- Small (~100 people) experiment searching for Neutrinoless Double Beta Decay
- Simple stack (Boost+ROOT+Geant4+Experiment), but low FTE on computing requires off-the-shelf solution
- Used Home/Linuxbrew for many years, reached limits of its “rolling release” and C++/Python support capabilities
- Spack identified through HSF as best tool going forward, both technically and to benefit from/contribute to community efforts
- Important to support Linux and macOS build-from-source
- No CVMFS hosting available to experiment, so binary packaging and/or Containers also required

Migrating SuperNEMO to Spack

- Migration system via fork of Spack on GitHub, plus custom `snemo` branch
  - Aim to support CentOS7, Ubuntu 18.04, macOS Mojave/Catalina natively, plus CentOS7 Docker/Singularity images
- Site-scope `packages.yaml` to reuse X11, GL, SSL, etc.
  - Same method as FCC
- Site-scope additional repository for SuperNEMO-specific packages and custom variants of certain Spack packages (e.g. Qt)
- Issues with C++ standard and macOS discussed and fixed upstream
- Now investigating use of metapackages and environments+views to create runtime/development environments
- Working with Key4hep on binary packaging, CVMFS deployment, and use with/over Docker/Singularity

Key4hep

- Software challenges are faced by detector community at future facilities
- Likely to be a Higgs-factory, but several different projects are possible:
  - CEPC, CLIC, FCC-ee, ILC
- Need for software which is robust, mature, yet sufficiently flexible to try new ideas



Jet tagging capabilities with 5TeV b-jets in FCC-hh, but using the CLIC software and the FCC vertex tracker, combined in the CLIC detector model

André Sailer, CLIC

See talk by André Sailer, Tuesday 17.45 Track X

Key4hep Prototype Build

- Build a software stack that can be used for key4hep workflows
  - Event generation
  - Simulation, with detector description
  - [Reconstruction], with experiment software framework
  - Analysis
- We selected to continue our work in Spack as the package orchestrator
  - Version 0.1
  - Spack first builds its own compiler (currently gcc9.2.0), for full self-consistency
  - Key top level packages:
    - Pythia, Geant4, DD4hep, Gaudi, ROOT
    - Use Spack's `packages.yaml` to set reproducible build options
  - *All building successfully*
  - Binary packages uploaded to build cache

Key4hep Prototype Build

- Installation is from build cache to new path
  - Same model as FCC
- Relocation is validated by checking the RPATH of relocated binaries and libraries
- Runtime environment is setup using environment modules
  - Commonly used in HPC centres
  - Sets up necessary entry point environment variables
  - Plus any auxiliary variables required by packages (e.g. Geant4 data files, ROOTSYS)
- N.B. Use of RPATH prevents interference between Key4hep stack and system binaries
- Basic tests in place to check functionality

Conclusions

- Building, packaging and deploying software is a shared problem across HEP
- HSF Packaging Working Group is an active open forum for discussion and cooperation
- Spack has been successfully tested as a build orchestrator for modern HEP software stacks
  - FCC
  - SuperNEMO
  - Key4hep
  - Neutrino experiments
- Production workflows now in development
  - Learning from FCC experience helps, switching to self-consistent Spack build actually makes this simpler

See SpackDev talk later this session