Open Source Build Tooling in High-Energy Physics Software

Case studies of Spack and CernVM-FS

This work benefited from support by the CERN Strategic R&D Programme on Technologies for Future Experiments (https://cds.cern.ch/record/2649646/, CERN-OPEN-2018-006).

Additional Thanks to Jakob Blomer, Radu Popescu, Gerardo Ganis and Graeme Stewart for various inputs.
HEP Software and Computing
Computing Workloads in HEP

- **Accelerator** vs. **Detector**
- **Online** vs. **Offline**

**Offline Data Processing:**
- Generation
- Simulation
- Digitization
- Reconstruction
- Analysis
Event Generation

\[
\sigma_{ab \to N} = \frac{1}{\text{cuts}} \int_0^1 d\Phi_N \frac{1}{2\xi} |M(\Phi_N, \mu_F, \mu_R)|^2
\]
Detector Simulation
Reconstruction
Analysis
The HEP Software Stack
Dependency Graph of HEP software Stack

Experiment-specific Packages
Dependency Graph of HEP software stack

Experiment-specific Packages + HEP-specific packages
Dependency Graph of HEP software stack

Experiment-specific Packages + HEP-specific Packages + General Purpose Libraries
Requirements for a Build System

- Need to be able to scale to a typical experiment software stack

A typical HEP stack contains some 300 packages
- 60 Experiment-specific
- 50 HEP-specific
- 200 System/General Purpose

14 GB install size, some 6h to build on single 4-core machine

- Combinatorics of multiple versions, platforms, Release/Debug ...

- Easy central deployment

- Reproducibility

- Support software development usecases
Build systems - previously...

- Often custom experiment-specific tools
- Fairly opaque, few transferable skills
- Difficult to maintain
- Need for a community-wide solution!
- Best practices in using build systems (Modern CMake ...)

13
Spack for HEP

- Originally written for/by HPC community
  - Emphasis on dealing with **multiple configurations** of the same packages
    - Different versions, compilers, external library versions ...
    - ... may coexist on the same system

- Not the only solution in this problem space:
  - EasyBuild
  - Nix/Guix
  - Conda
  - Gentoo Prefix

See [https://hepsoftwarefoundation.org/notes/HSF-TN-2016-03.pdf](https://hepsoftwarefoundation.org/notes/HSF-TN-2016-03.pdf) for a comparison

And: previous FOSDEM talks about Spack by T. Gamblin!
Spack packages are simple Python

```python
from spack import *

class DyninstPackage:
    """API for dynamic binary instrumentation. Modify programs while they
    are executing without recompiling, re-linking, or re-executing."""
    homepage = "https://paradyn.org"
    version('8.2.1', 'dbf60b7fa6ade7a2e8', url='http://www.paradyn.org/release8.2.1/DyninstAPI-8.2.1.tgz')
    version('8.1.2', 'b0f83b33755a0a661', url='http://www.paradyn.org/release8.1.2/DyninstAPI-8.1.2.tgz')
    version('8.1.1', '02862955b3760a7b9', url='http://www.paradyn.org/release8.1.1/DyninstAPI-8.1.1.tgz')
    depends_on('libelf')
    depends_on('libdwarf')
    depends_on('boost@1.42.2')

    # new version uses cmake
    def install(self, spec, prefix):
        libelf = spec['libelf'].prefix
        libdwarf = spec['libdwarf'].prefix
        with working_dir('spack-build', create=True):
            cmake('..',
                  '-DBOOST_INCLUDE_DIRS=%s % spec['boost'].prefix.include,
                  '-DBOOST_LIBRARY_DIRS=%s % spec['boost'].prefix.lib,
                  '-DBOOST_NO_SYSTEM_PATHS=TRUE',
                  'DLIBELF_INCLUDE_DIRS=%s % join_path(libelf.include, ['libelf'],)',
                  'DLIBELF_LIBRARIES=libelf % join_path(libelf.lib, ['libelf.so'],)',
                  'DLIBDWARF_INCLUDE_DIRS=%s % join_path(libdwarf.include, ['libdwarf'],)',
                  'DLIBDWARF_LIBRARIES=%s % join_path(libdwarf.lib, ['libdwarf.so'],)
                  *std_cmake_args)
            make()
            make('install')

    # Old version uses configure
    @run_after('@8.1.1')
    def install(self, spec, prefix):
        configure("--prefix=" + prefix)
        make()
        make('install')
```

Metadata
Version/URLs
Dependencies
Patches (not shown)
Commands for install
Access build config through spec.
Spack basics

- Spack handles combinatorial version complexity by assigning hashes /a83xd43
- Concretization fills in missing configuration details when the user is not explicit

```bash
spack install root
spack install root@6.20.04
spack install root@6.20.04 % gcc@9.3.0
spack install root@6.20.04 % gcc@9.3.0 target=broadwell
spack install root@6.20.04 % gcc@9.3.0 ^python@3.8.2
```
Status of HEP software in Spack

- Around 80 package recipes with `hep` tag
  - pythia8 photos syscalc madgraph5amc g4emlow lcio heputils vbfnlo g4inc1 lhapdf sherpa py-particle openloops herwigpp heppdt tauola lhapdfsets root collier garfieldpp py-uproot geant4 evtgen clhep gaudi geant4-data vgm alpgen relax g4tendl g4realsurface genfit py-hepdata-validator g4nd1 cool recola fjcontrib delphes dd4hep whizard hepmc3 pythia6 g4enadfstate coral g4photonevaporation kassiopeia yoda acts chaplin simsipm mcutils recola-sm py-hepunits herwig3 hepmc py-gosam ccs-qcd g4saiddata g4neutronxs aida g4radioactivedecay podio edm4hep g4abla njet apfel fastjet qd thepeg rivet vecgeom gosam-contrib unigen hepmcanalysis geant4-vmc genie g4pii py-uproot4 hoppet qgraf g4particlexs dire

- Many (> 25) active maintainers from different experiments
- 3rd party repositories for full experiment stacks
  - EIC: [https://github.com/eic/eic-spack](https://github.com/eic/eic-spack)
  - Key4hep: [https://github.com/key4hep/key4hep-spack](https://github.com/key4hep/key4hep-spack)
  - LCG-releases: [https://gitlab.cern.ch/sft/sft-spack-repo](https://gitlab.cern.ch/sft/sft-spack-repo)
Open Issues and Wishlist

- Data packages
  - Implies compiler dependency
- Avoid rebuilding ROOT! (thank you for `spack install --reuse`)
- Building glibc
- Nightly builds from git master
- Relocation and build path
- Setup script generation
Software Deployment with CVMFS
Many workloads in HEP are embarrassingly parallel!

Off-line, single events can usually be processed on modest hardware
  - Nevertheless large volume - truly Big Data!
  - Distributed computing and “Computing Grid”

Still much potential to use heterogeneous computing and HPC resources.
The LHC Computing Grid

Started in 2002
Provide processing power and data access to physicists
~170 centres in 42 countries
Running 24/7/365
The Anatomy of a HEP Software Stack

Key Figures for LHC Experiments
- Hundreds of (novice) developers
- > 100 000 files per release
- 1TB / day of nightly builds
- ~100 000 machines world-wide
- Daily production releases, remain available “eternally”
Software distribution - CernVM-FS

- Read-only, globally distributed file system optimized for software delivery
- Provides uniform, consistent and versioned POSIX file system access to /cvmfs

```
$ ls /cvmfs/cms.cern.ch
slc7_amd64_gcc700  slc7_ppc64le_gcc530  slc7_aarch64_gcc700  slc6_mic_gcc481 ...
```

- Populate and propagate new and updated content
  - A few “software librarians” can publish into /cvmfs
  - All content in /cvmfs is cryptographically signed
  - Transactional writes as in git commit/push
- More details under [https://cernvm.cern.ch/fs/](https://cernvm.cern.ch/fs/)
CVMFS Deployment

Several possible workflows (see presentation by Jakob Blomer)

1. The Postscript relocation approach
2. The rsync approach
3. The Gateway approach
4. The Container approach

The rsync approach:

- Builder mounts a read/write copy of the /cvmfs tree
- Builder changes/installs software in place
- Publisher uses rsync to pull changes from the builder

Non-trivial to maintain multiple, synchronized publishers
Conclusions

High Energy Physics faces some fairly unique challenges...

- And innovated on tools to solve them (see CVMFS in this presentation)

... but also many common ones, and would probably not be possible without the wider OSS ecosystem.

- **HEP 💙 Open Source!**
  - Open Data projects
  - Software Projects:
    - ROOT
    - Geant4
    - Indico
    - ...

Future Collider Projects are collaborating on Software (Key4hep)!

FCC, CLIC, ILC, CEPC, ...