A New Build System for LArSoft

Brett Viren

Physics Department

BROOKHAVEN NATIONAL LABORATORY

LArSoft Libbers

16 Dec 2014

Outline

Why a new build system?

What's wrong with the current system?

Overview of new build system

Status

Integration Plan

Pertinent LBNE S&C Requirements[1]

- LBNE software stack must build on LBNE platforms in an automated fashion.
- Compile 3rd party s/w from pristine sources + known patches.
- Retain provenance of software, reproduce releases.
- LBNE must determine contents/versions of released LBNE s/w stack.

[1] LBNE DocDB 8035

Primary problem:

Inverted dependency art-based packages build with CMake using custom macros that depend on local-Fermilab infrastructure (UPS).

Other issues:

- Significantly burdens building of the software stack.
- Effectively impossible to port to new platforms, blocks contributions from many in the collaboration.
- Release definition not in direct LBNE control, tied to larger experiment landscape.
- Does not build all LBNE software.
- Leads to a painful development environment.

→ All this for a code base (art+LArSoft)
  - which is just ∼1/5 of just ROOT.

→ But, an incredibly critical code base for LBNE!

→ Continued support from Fermilab SCD also critical!

Parts to the pure-CMake build

package level pure-CMake, modern, best practices and free of UPS entanglements.

build config suite-wide configuration of all build parameters in simple text files.

orchestration automated building of all packages in the stack.

runtime Support for UPS and other runtime environments.

Pure CMake build

What is it?

- Provide new `CMakeLists.txt` files for art-based packages.
- Standard CMake patterns, no dependency on local-Fermilab vagaries (eg UPS/cetbuildtools).
- The "installer" (person/code running `cmake`) must tell where to find dependencies and rely on CMake built-in detection.

That is, do things in the normal CMake way:

```
$ cmake -DCMAKE_CXX_COMPILER=/path/to/bin/g++ \
        -DSQLite3_LIBRARY=/path/to/lib/libsqlite3.so \
        [....]
$ make && make install
```

Build Configuration Management

the good build configuration control in hands of the installer.

the bad see "the good"

A simple language ("worch") captures build configuration information in a platform independent manner.

```
[group externals]
packages = [...], sqlite, [...]
environment = group:buildtools, group:compiler

[package sqlite]
version = {sqlite_version}
source_url = http://www.sqlite.org/2014/{package}-autoconf-{sqlite_version_zeros}.tar.gz
source_unpacked = {package}-autoconf-{sqlite_version_zeros}
unpacked_target = configure
build_target = sqlite3
install_target = bin/sqlite3
export_PATH = prepend:{install_dir}/bin
```

Placing these configuration files in a repository provides trivial suite-wide release definition management.

Build Orchestration

LBNE LArSoft stack: $>$ 40 packages, $>$ 500 build steps.

- Automate and parallelize to make full use of beefy build hosts, not use human time.
- Interpret "worch" configuration files into "waf" tasks.
  - Waf is a powerful, Python-based low and high level build tool.
  - Mature and actively developed/maintained.
  - Used for some big names such as Samba.

One command to build them all:

```
$ waf --prefix=/path/to/install \
      --orch-config=lbne/suite-larsoft.cfg \
        configure build
```

Build Machinery Packaging

Python ecosystem heavily leveraged:

- Python packaging:
  - `worch` configuration + waf bindings
  - `ups-utils` general Python modules for managing biary UPS products areas (includes `urman` CLI)
  - `worch-ups` Waf tools for producing UPS binary packs.
  - `lbne-build` LBNE worch configuration + special waf tools.
- Python Package Index (PyPI) registration/uploads

To set up all build machinery is not much more than:

```
$ pip install lbne-build
```

End-user Runtime Support

Many different runtimes may be supported at once.

- Runtime conventions expressed as Worch configuration.
- Environment and dependencies transferred from Worch files to runtime-specific configuration files.
- Current support for **Environment Modules** and **UPS**
  - UPS binary packages are being produced now
  - Investigating novel packaging such as Conda/Binstar

→ At the end of the day, one may still have a "Fermilab compatible" **UPS products area**.

→ Plus, ability to evaluate and support **novel paradigms** contemporaneously.

Center of Development

LBNE GitHub organization:

https://github.com/LBNE/

- Contains **tracking** forks of art and LArSoft packages.
  - Can merge "upstream" development.
  - Can push GitHub $\rightarrow$ Redmine/git.
- Used GitHub to **avoid interference** with ongoing use during initial development phase.
- Mostly **just CMake files modified**, some bugs found/fixed in actual code (with Redmine tickets issued).
- LBNE GitHub org also holds LBNE Worch configuration package:

https://github.com/LBNE/lbne-build/

Build status

What works:

- Fully automated build of externals + art + LArSoft + `lbnecode` and production of UPS binary packs.
- Platforms
  - **SL 6.4**, full support + Jenkins builds
  - **Ubuntu 14.04**: full support (Jenkins slaves needed)
  - **Mac OS X** (GCC): support + Jenkins in progress
    - → GCC'isms in art blocking CLANG support
  - **Fedora Core 20**: initial support (for ICARUS s/w devs)

Still to do (immediate)

- More testing of general build results and specifically production of UPS binary packs.
- Apply "integration plan" (next slide) to `lbnecode` package in Redmine/git.

Integration Plan

A written plan exists[2] and presented to SCD:

- Add pure-CMake build to art-based packages in Redmine/git.
- Leave existing UPS/CET build system as-is and as default.
- Option to switch to pure-CMake build.
- Apply this to `lbnecode` first.
  - Present findings of this "port" to LArSoft project
  - Discuss applying "port" to LArSoft packages
- Depending on success, repeat with art project
- Evaluate/exercise pure-CMake build by CET
- Flip switch to make default.

→ **Lots of discussion, negotiation, acceptance still needed**.

[2] Add LBNE DocDB

Principle Instigators

An incomplete list:

Ben Morgan Initial pure-CMake package build method and their application to art.

Patrick Gartung Application of pure-CMake to LArSoft and `lbnecode`, Jenkins integration, Worch configuration development, Ubuntu and Mac porting.

Brett Viren Worch and initial `lbne-build` configuration, UPS packaging.