

LBNE Software Discussion, Build system and Larsoft Releases

Brett Viren

Physics Department



May 2013

Outline

LBNE Software Build System

- Goals for LBNE Software Build System

- Scope LBNE Software

- LBNE Computing Platforms

- Build System Improvements

- Porting Problems

Larsoft

- Questions, status reports

- Defining Larsoft Releases

- Managing User Environment

- Continuous Building

Goals for LBNE Software Build System

- **Overall strategy:** leverage existing effort/systems currently in use at FNAL, fold back in any improvements.
- Build all (non-system) packages needed by “the experiment software”.
- Build releases (frozen, development, personal) of the experiment software
- Support “major” (t.b.d.) LBNE collaboration platforms.
- User environment management.
- Support source and binary installations .
- Specify package versions independently from other experiments.
- Automate build and maintain a continuous build system on all supported platforms.

Scope of LBNE Software

My hope is that we unify the software effort across LBNE as much as possible. This includes:

- Offline simulations, reconstruction, geometry modeling (larsoft, ...)
- Beam simulations (g4lbne, ...)
- Online/DAQ/RC/SC/Monitoring (35t and final far detector)
- Sensitivity calculations (globes, ...)
- Long way away but beam monitoring data (NG of MINOS's beam-data-process: IFBeam)

Almost all of this effort is off-project and thus the collaboration's responsibility. I want to leverage software effort, where beneficial, from other LAr experiments at FNAL and elsewhere (eg, CAPTAIN and other new LBNE-related software initiatives)

LBNE Software and Computing Platforms

- Supported set driven by LBNE Collaborations needs and platform existence.
- Every platform needs one or more “champions”:
 - Officially identified (eg, on a list on a Redmine wiki page)
 - Committed to assuring platform remains supported
 - Performs initial port of LBNE software and externals
 - Respond to platform-specific problems and requests for help by newbies
 - Fixes platform-specific failures in the build
 - Maintains a client for the continuous builder (BuildBot)
- Will poll collaboration to determine initial set.

Leveraging Existing FNAL Build System

- Retain UPS for user environment management
 - UPS'ify larsoft – `larsoft.table` file started
- Want to produce and distribute UPS-compatible binary packages
 - But, still provide from-source build automation (more on this).
 - Work out how other platform “champions” can push their binary builds to central distribution server at FNAL.
- Use FNAL distribution servers for source and binaries
- Hope to integrate desired changes to current FNAL workflows to “exploit” existing effort
 - I expect changes to actually ease build-related workloads.
- Follow developments with CVMFS

New Directions in the build system

Basic ideas:

granular break monolithic tarballs, atomic to the package level

regular design high-level build orchestration

pristine mirror unmodified upstream source tarballs for convenience, or allow download from upstream by installer

abstract define build orchestration protocol, write per-package shim scripts to fit

version manage/distribute all shim scripts and build orchestration in/from a code repository

strict loudly aborts on failure, no quiet/ignored errors

idempotent rerun all or part of a build step and not repeat steps that already successfully completed

Want this command to exist:

```
$ orchestrate <suite-name> <suite-version>
```

Prototype LBNE beanie

- Developed as a set of bash scripts during the port of larsoft to RACF's SL5.3
- Builds on ideas used in MINOS (msrt), Daya Bay (dybinst), LBNE WCD (garpi) and others (metascons)
- Prototype: somewhat quick and dirty, implements the ideas but not something to use forever.
- Leverages SVN's `svn:externals` to define LBNE experiment software releases.
- Calls the various `build*.sh` scripts.

Beanie: <https://cdcvs.fnal.gov/redmine/projects/lbne-software/repository/show/build/scripts>

Metascons (experimental):

<https://github.com/brettviren/metascons>

Using beanie

Instructions at: https://cdcvs.fnal.gov/redmine/projects/lbne-software/wiki/Installation_From_Source_with_beanie

```
svn checkout http://cdcvs.fnal.gov/subversion/lbne-software/build
./build/scripts/beanie-larsoft development lbnesoft
```

- Downloads the art* and nue_extras tarballs from FNAL.
- Downloads Larsoft+nusoft (and SRT) from SVN via lbne-software Redmine project.
- Applies patches, builds everything, installs larsoft.table.

Desired changes to current build system

- Develop a high level build orchestration system.
 - `metascons` has promise,
 - but a cleaned up `beanie` approach an alternative
- Break existing source tarballs into:
 - pristine, per-package source tarballs
 - separate tarball holding all build orchestration files
 - but have these produced from tagged code repository
- Rewrite existing `build*.sh` files, rely on shared library of reusable functions
- Start Redmine instance and repository to hold all this.

These are largely orthogonal efforts but need coordination.

Problems found during RACF SL5.3 64bit build

General issues:

- Manual instructions with various small inconsistencies make for easy mistakes.
- Small documentation errors and bad assumptions led me down some wrong paths.
- Things were (always are!) in a state of flux so documentation/reality mismatch and changing build qualifiers.

Specific snags:

- RACF has multiple system GCCs installed which foiled locating GCC/libstdc++ files during the build of GCC 4.7.1.
- The string "c1" in the directory `.../clean/...` caused GCCXML to think it was building on a MicroSoft platform(!)
- Broken larsoft checkout script (nusoft packages silently fail to checkout) (pre-UPS'ified nusoft)

Problems found on Debian/Ubuntu

Initial port to non-SL:

- GCC and Debian multiarch do not agree on file locations. Fixed with some special GCC build environment variables.
- Not really Deb/Ubu but found ART does not build on 32-bit systems during this port.

Port is ongoing on new Ubuntu 12.04 LTS 64bit workstation at BNL.

Questions on the split of Nusoft

My build of Larsoft on RACF used Nusoft pulled from SVN.

- Has nusoft now been UPS'ified?
- Can someone give a summary and status?
- Does it still rely on SRT? Does a .table file exist?

Questions on move larsoft from SRT to CMake

I hear “rumors” that Larsoft will be converted from SRT to CMake

- True? Status?
- Does it still allow for the concept of “public” and “private” releases
- I have a `larsoft.table` file but it needs work, has anyone started on providing one?

Larsoft release definition

- Larsoft releases are currently defined in text files, one for each release
- This is taken from MINOSSOFT where SRT and CVS limitations required this
- More modern repositories (eg, SVN) allow directly for release definition

Suggestions follow.

Release definitions with SVN

Two common methods:

- `svn copy` directories to `tags/<tag>/` or `branches/<branch>/` subdirectory.
 - These are just arbitrary labels.
 - In both cases they are really branches.
 - New revisions can be easily committed so they are not safely point releases
 - Suitable when a branch is really intended.
- `svn:externals` act kind of like file system “symlinks”
 - Can link to any SVN URL, even on a different server
 - Links can specify an exact SVN revision number
 - On checkout, these links are followed and user gets the “real” files
 - Suitable when a true point release is intended.

Larsoft environment setup

Currently:

- There is one loooooong if/elif/elif/elif/else/fi branch chain in the larsoft setup script.
- This chain will grow for each subsequent release.
- It's overly verbose (it specifies some UPS packages that will get pulled in anyways)

The fix:

- Define a UPS `larsoft.table` file and users do `setup larsoft` development like any UPS package.
- Keep this `.table` file in with larsoft source code so it is version controlled with new version for each release
 - This is how UPS is supposed to be used
 - No ever growing if/elif/elif/else/fi chain.

Initial stab at this `larsoft.table` exists but could probably use some UPS expert help.

Continuous Building

- In Daya Bay we very successfully use “bitten”, a continuous builder with ties to the Trac bug tracker.
- Changes are checked in to SVN, after a quiet period (1 hour) the bitten “master” tells the “slaves” to start a new build. Results are collected and can be browsed from the Trac web server.
- I've been looking at setting up BuildBot which is similar to bitten but bug tracker independent.

Rick Snider:

Just wanted to note that MINERvA is using buildbot for nightly builds. SCD is providing support for that and will do so here if desired.

- I want to pursue using this resource
 - Who is the technical contact? (A: expert is Marc Mengel, Rick is contact)I guess there is a myriad of technical details to work through.
 - Start with BuildBot client on RACF building Larsoft+ART+externals
 - Add G4LBNE next.