Removing more header and library dependencies

Kyle J. Knoepfel

13 August 2019

LArSoft coordination meeting

Continuing the quest

- Trying to remove unnecessary dependencies from LArSoft.
- Reasons
  - **Ease the maintenance burden**
  - Faster compilation and link-times
  - Smaller libraries
  - Smaller run-time overhead
- I have presented ways to cleanup code in the past

- Today, I will show the results of removing unnecessary headers and link-time dependencies.

include-what-you-use.org

The main goal of include-what-you-use is to remove superfluous #includes. It does this both by figuring out what #includes are not actually needed for this file (for both .cc and .h files), and replacing #includes with forward-declares when possible.

- iwyu relies on Clang to do the static analysis
- It's a bit cumbersome to use in a UPS environment, but I hacked something together that worked.
- Not all of the reports are correct, so you have to use some judgment.
- I don't show details of the tool, just the consequences of "obeying" it.
- Can find a way to make this usable in LArSoft if there is a desire to do so.

Removing unnecessary library dependencies

- CMake provides a link-what-you-use feature, which will report unnecessary library dependencies.
- The way we build things makes it difficult to use this feature in practice.
- I went for the brute-force solution:
  - Remove all library dependencies in the CMakeLists.txt files, and incrementally add the required dependencies until there were no more link-time failures.

- This library-pruning is unlikely to result in large savings in build times or built object sizes.
  - The savings is in maintenance and in reduce run-time overhead.

Consequence of pruning

- Baseline is LArSoft v08_27_02
- Lines of code
  - Released: 348079
  - Pruned: 345934
- Build time (c2:prof and 24 cores)
  - Released: 11 min 5 secs
  - Pruned: 10 min 2 secs
- Size of installed libraries (c2:prof)
  - Released: 1.93 GB
  - Pruned: 1.88 GB
- Run-time improvements
  - Depends on workflow

Consequence of pruning

- Baseline is LArSoft v08_27_02
- Lines of code
  - Released: 348079
  - Pruned: 345934
- Build time (c2:prof and 24 cores)
  - Released: 11 min 5 secs
  - Pruned: 10 min 2 secs
- Size of installed libraries (c2:prof)
  - Released: 1.93 GB
  - Pruned: 1.88 GB
- Run-time improvements
  - Depends on workflow

Package	Version	Released	Pruned
larana	v08_10_06	**171MB**	**166 MB**
larcore	v08_04_07	17 MB	17 MB
larcorealg	v08_14_00	23 MB	23 MB
larcoreobj	v08_05_01	1.4 MB	1.4 MB
lardata	v08_07_02	**129 MB**	**122 MB**
lardataalg	v08_08_02	18 MB	17 MB
lardataobj	v08_04_07	**155 MB**	**149 MB**
lareventdisplay	v08_08_06	83 MB	77 MB
larevt	v08_06_04	49 MB	49 MB
larexamples	v08_02_13	41 MB	40 MB
larg4	v08_03_11	24 MB	22 MB
larreco	v08_16_03	**813 MB**	**772 MB**
larsim	v08_11_01	**247 MB**	**240 MB**
larwirecell	v08_05_11	39 MB	38 MB
Total		**1.93 GB**	**1.88 GB**

Feature branches

Upstream repositories

artg4tk	develop (new tag required)

LArSoft repositories

larana	feature/knoepfel_rm_unused_headers
larcore	feature/knoepfel_rm_unused_headers
larcorealg	feature/knoepfel_rm_unused_headers
larcoreobj	feature/knoepfel_rm_unused_headers
lardata	feature/knoepfel_rm_unused_headers
lardataalg	feature/knoepfel_rm_unused_headers
lardataobj	feature/knoepfel_rm_unused_headers
lareventdisplay	feature/knoepfel_rm_unused_headers
larevt	feature/knoepfel_rm_unused_headers
larexamples	feature/knoepfel_rm_unused_headers
larg4	feature/knoepfel_rm_unused_headers
larreco	feature/knoepfel_rm_unused_headers
larsim	feature/knoepfel_rm_unused_headers
larwirecell	feature/knoepfel_rm_unused_headers

Experiment repositories

ubevt	feature/knoepfel_rm_unused_headers
ubreco	feature/knoepfel_rm_unused_headers
ubsim	feature/knoepfel_rm_unused_headers
lariatsoft	feature/knoepfel_rm_unused_headers
dunetpc	feature/knoepfel_rm_unused_headers
sbndcode	feature/knoepfel_rm_unused_headers