Compression of raw digits

Gianluca Petrillo

`LArSoft` Librarians' meeting, May 20th , 2014

Memory footprint of data from ROOT tree

ROOT trees with `art` are a complicate business.

Each data product read from or written to a ROOT tree uses some of the following:

- ROOT "baskets" to read compressed data from the tree
- uncompressed objects where input data is read from the tree
- object in the event (is that the same as above or below?)
- uncompressed objects from which output data is written to the tree
- ROOT "baskets" to write compressed data to the tree

This can effectively double or triple the memory requirement of a product. And tree buffers are permanent[1]: if an event requires 100 MB of buffers, those buffers will survive for many events after that one.

[1] Exceptions apply.

A reco3D output file

An indication of where to look can be given by the size of the data in the tree. From a µBooNE 3.6 GB output (ν + cosmics, down to 3D reco), 5 events:

`recob::Wire`, 1.6 GB already addressed by Baller's "regions of interest" approach

`raw::RawDigit`, 790 MB produced during detector simulation

`sim::SimPhotons`, 475 MB produced during Geant4 simulation

`simb::MCParticles`, 285 MB also from Geant4 simulation

`optdata::ChannelDataGroup`, 245 MB from optical detector simulation

Each of these is a good candidate for investigation:

Can the data in these object be reduced?

Today's study: `raw::RawDigit`

- contains a list of ADC counts (16-bit each)
- produced by the detector or the detector simulation (`SimWireXxxx`)
- supports data compression (Huffman already implemented)
- data compression is turned off by default

Well, today I lift my finger just enough to enable that compression:

```
physics.producers.daq.CompressionType: Huffman
```

Today's study: `raw::RawDigit` tests

The following test has been performed with `standard_detsim_3window_uboone.fcl` (wire and optical digitization and output) on a ν_e + cosmic rays, 100-event sample.

`RawDigit` compression	`none`	`Huffman`
file size	6121 MB	6061 MB
`RawDigit` branch size	274 MB	213 MB
data size	37.98 GB	22.56 GB
`RawDigit` data size	15.87 GB	0.45 GB
total time	2195 s	2144 s
`SimWireXxxx` time	1115 s	1188 s
`RootOutput` time	735 s	550 s
peak total memory[2]	2.3/2.9/3.1 GB	1.6/2.5/2.8 GB
`RootOutput` memory	795/941/1000 MB	415/561/620 MB

(the file sizes in the first block are from the Reco3D test in the previous page)

`RootOutput` compression level is the default (`7`).

[2] Sampled at three different times. `RootOutput` memory monotonically increases.

Conclusions

- the size of data products heavily affects memory usage
- reducing this size scales into a large reduction of used memory
- I am going to investigate the most apparent cases looking for solutions
- reduction in memory size usually means need of more CPU to use the products
- compression of `raw::RawDigit` has quite some advantage...

Is the code ready for `raw::RawDigit` compression?

`raw::RawDigit` does not provide an automatic way to compress/uncompress the digits. The user of `raw::RawDigit` *must* uncompress the data each time before using it.
Is all the code following that prescription?