Profiling of LBNE code

Gianluca Petrillo

LArSoft stakeholders' meeting, February 12th , 2014

Introduction: the speaker

Hello!

I spent my last years doing physics:

E835	$c\bar{c}$ (charmonium) bound states from $\bar{p}p$ fix target experiment (FNAL)
CMS	Higgs analysis study in Vector Boson Fusion production (2007: simulation!)
DØ	Jet Energy Scale calibration, top quark mass measurement
	... and I still use LaTeX for talks

I have started as a consultant in the Computing Division on January, to provide help (and some noise) on `LArSoft`.

Profiling LArSoft

Large part of my time is being devoted to the optimization of the LArSoft code:

computing time because faster is better (and sometimes cheaper)

resources to fit in batch/grid environments with restricted use of resources

Profiling LArSoft

My first task:

Study of the use of memory of LBNE code

- a lot of worker nodes allow no more than 2 GiB of memory, all included
- a simple one-event simulated with the full LBNE detector may require more than 6 GiB

⇒ that is sometimes called “a problem”

- while LBNE is chosen for its blatancy, µBooNE should also benefit from a solution
- as I gain expertise, I will also work for optimizations of speed

Profiling tools – timing

- `fast` works fine, big text files as output
- `callgrind` (`valgrind` tool) has some visualization tools available; I haven't tried it yet
- `gperftools` works just as well, provides (questionable) time break-up inside functions, outputs in `callgrind` format
- `IgProf` needs code changes, but it can snipe at a specific part of the program; I haven't tried it yet
- `Open|SpeedShop` was a pain to compile, and I can't make it work yet; but very appealing
- `Timing` (`art` service) provides a first direction

Time profilers are usually fast to run (e.g. `fast` has a overhead of less than 5% as adesign guideline.

Profiling tools – memory

`massif` (`valgrind` tool) works nice and has some visualization tools available; the KDE4 one relies on a (currently broken) KDE “dot” viewer

`DHAT` (`valgrind` tool) not tried yet, no visualization helper found

`IgProf` not tried yet

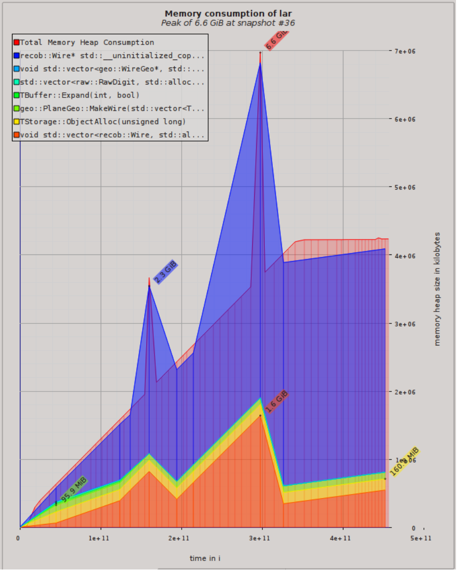
`TotalView` is available and working from UPS; it’s mainly a debugger: great for seeing the culprit performing the crime live

`SimpleMemoryCheck` (`art` service) mostly useful to detect large memory leaks

Memory tracking makes the runs become very time-consuming.

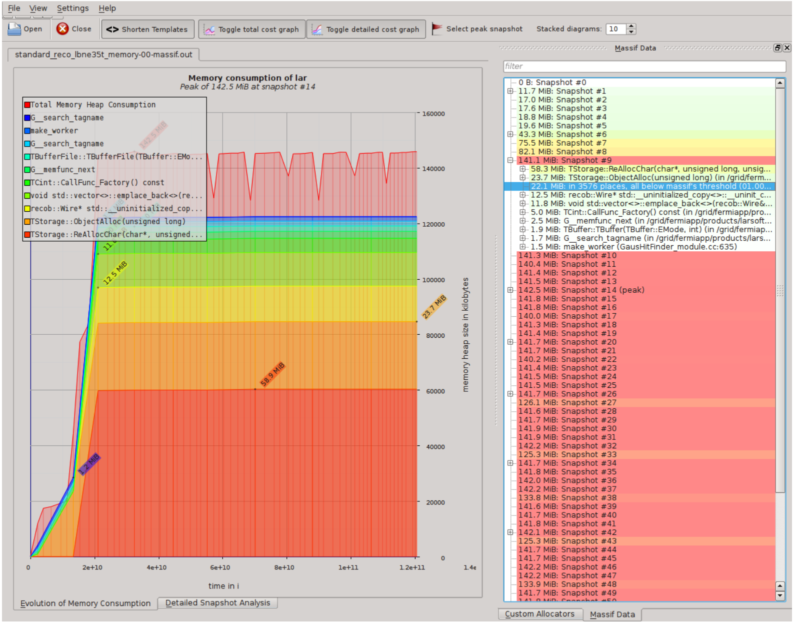
An example: LBNE event generation

Showing here the generation: `prodsingle_lbnefd.fcl` (10 events)



We can identify a transient peak up to 6.6 GiB *and the code which allocated it* (not shown: `CalWireLBNE10kt::produce()`; the reference to the source code points to `stl_vector.h:920`, mislead by compiler optimization). Also note that the memory usage, beside the peak, is still too large (> 4 GiB).

LBNE 35 tons: standard reconstruction


File
View
Settings
Help
Open
Close
<> Shorten Templates
Toggle total cost graph
Toggle detailed cost graph
Select peak snapshot
Stacked diagrams: 10
standard_reco_lbne35t_memory-00-massif.out
Memory consumption of lar
Peak of 142.5 MiB at snapshot #14
Total Memory Heap Consumption
G__search_tagname
make_worker
G__search_tagname
TBufferFile::TBufferFile(TBuffer::EMo...
G__memfunc_next
TCint::CallFunc_Factory() const
void std::vector<>::emplace_back<>(re...
recob::Wire* std::__uninitialized_cop...
TStorage::ObjectAlloc(unsigned long)
TStorage::ReAllocChar(char*, unsigned...
142.5 MiB
12.5 MiB
23.7 MiB
58.9 MiB
1.2 MiB
160000
140000
120000
100000
80000
60000
40000
20000
0
memory heap size in kilobytes
0
2e+10
4e+10
6e+10
8e+10
1e+11
1.2e+11
1.4e
time in i
Evolution of Memory Consumption
Detailed Snapshot Analysis
Massif Data
filter
Custom Allocators
Massif Data

```
0 B: Snapshot #0
11.7 MiB: Snapshot #1
17.0 MiB: Snapshot #2
17.6 MiB: Snapshot #3
18.8 MiB: Snapshot #4
19.6 MiB: Snapshot #5
43.3 MiB: Snapshot #6
75.5 MiB: Snapshot #7
82.1 MiB: Snapshot #8
141.1 MiB: Snapshot #9
  58.3 MiB: TStorage::ReAllocChar(char*, unsigned long, unsig...
  23.7 MiB: TStorage::ObjectAlloc(unsigned long) (in /grid/ferm...
  22.1 MiB: in 3576 places, all below massif's threshold (01.00...
  12.5 MiB: recob::Wire* std::__uninitialized_copy<>::__uninit_c...
  11.8 MiB: void std::vector<>::emplace_back<>(recob::Wire&...
  5.0 MiB: TCint::CallFunc_Factory() const (in /grid/fermiapp/pro...
  2.5 MiB: G__memfunc_next (in /grid/fermiapp/products/larsoft...
  1.9 MiB: TBuffer::TBuffer(TBuffer::EMode, int) (in /grid/fermiap...
  1.7 MiB: G__search_tagname (in /grid/fermiapp/products/lars...
  1.5 MiB: make_worker (GausHitFinder_module.cc:635)
141.3 MiB: Snapshot #10
140.4 MiB: Snapshot #11
141.4 MiB: Snapshot #12
141.5 MiB: Snapshot #13
142.5 MiB: Snapshot #14 (peak)
141.8 MiB: Snapshot #15
141.8 MiB: Snapshot #16
140.0 MiB: Snapshot #17
141.3 MiB: Snapshot #18
141.4 MiB: Snapshot #19
141.7 MiB: Snapshot #20
141.7 MiB: Snapshot #21
140.2 MiB: Snapshot #22
141.4 MiB: Snapshot #23
141.5 MiB: Snapshot #24
141.5 MiB: Snapshot #25
141.7 MiB: Snapshot #26
126.1 MiB: Snapshot #27
141.6 MiB: Snapshot #28
141.7 MiB: Snapshot #29
141.9 MiB: Snapshot #30
141.9 MiB: Snapshot #31
142.2 MiB: Snapshot #32
125.3 MiB: Snapshot #33
141.7 MiB: Snapshot #34
141.8 MiB: Snapshot #35
142.0 MiB: Snapshot #36
142.2 MiB: Snapshot #37
133.8 MiB: Snapshot #38
141.6 MiB: Snapshot #39
141.7 MiB: Snapshot #40
141.8 MiB: Snapshot #41
142.1 MiB: Snapshot #42
125.3 MiB: Snapshot #43
141.7 MiB: Snapshot #44
141.7 MiB: Snapshot #45
142.2 MiB: Snapshot #46
142.2 MiB: Snapshot #47
133.9 MiB: Snapshot #48
141.7 MiB: Snapshot #49
141.8 MiB: Snapshot #50
```

Other test runs

The following test runs used 10 events, `LArSoft` and lbnecode based on `v1_00_02`.

Configuration file	profiler	time/event	peak memory
`prodsingle_lbne35t.fcl`	none	2.5”	0.8 GiB
	`massif`	133”	78 MiB
`standard_reco_lbne35t.fcl`	none	4.7”	0.7 GiB
	`massif`	169”	140 MiB
`prodsingle_lbnefd.fcl`	none	55”	2.3 GiB
	`massif`	3000”	1.0 GiB
`standard_reco_lbnefd.fcl`	none	aborted	3.4 GiB
	`massif`	aborted	6.6 GiB
`prodsingle_uboone.fcl`	none	16”	1.4 GiB
	`massif`	725”	0.5 GiB
`standard_reco_uboone.fcl`	none	20”	1.5 GiB
	`massif`	200”	0.6 GiB

All runs include always `SimpleMemoryChecker` and `Timing` `art` services. Plain and `massif`’d runs use different random seeds.

What's next

This is just a quick view of the tools to achieve a goal.
My next steps:

- spend the rest of the week to familiarize with the tools
- then start the real work:
    - get a better idea of what the code is doing
    - interact a lot with the authors to understand the code, its design
    - a test unit would be *very* useful to validate any candidate fix

Thomas Junk has pointed me to a couple of configuration files which blow up the memory. *Other pointers are very welcome*.

Additional material

Crashed??

This was not fatal:

```
%MSG-w HitCheater:  HitCheater:hitcheat 10-Feb-2014 14:13:00 CST  run: 1 s
caught exception
---- Geometry BEGIN
  Can't find Cryostat for position (nan,nan,nan)
---- Geometry END
when attempting to find TPC for position move on to the next sim::IDE
```

Crashed!

```
~~~~~~~~~~~ Running Disambiguation ~~~~~~~~~~~

APA 0:
  Trivial Disambig -->  92 / 365 U,  144 / 377 V
  Crawl            -->  365 / 365 U,  377 / 377 V
          Found 4 endpoint hits in apa 0
           endP on channel 0 at time 1844.07
           endP on channel 511 at time 1846.01
           endP on channel 1200 at time 1858.96
           endP on channel 1988 at time 1862.28
Zcent = 251.767, UVintersects zpos =
TimeModule> run: 1 subRun: 0 event: 1 apahit APAHitFinder 2.01072
TimeEvent> run: 1 subRun: 0 event: 1 164.511
%MSG-s ArtException:  PostPathEndRun end_path 10-Feb-2014 23:01:07 CST  Po
cet::exception caught in art
---- EventProcessorFailure BEGIN
  An exception occurred during current event processing
  ---- ScheduleExecutionFailure BEGIN
    ProcessingStopped.

    ---- ThreeChanPos BEGIN
      U/V channels don't intersect, bad return.
      cet::exception going through module APAHitFinder/apahit run: 1 subRu
    ---- ThreeChanPos END
```