LArSoft - Bug #17708

SimPhotonCounter causes significant memory usage for DUNE library generation.

09/13/2017 09:44 PM - Jason Stock

Status:	Closed	**Start date:**	09/13/2017
Priority:	High	**Due date:**	
Assignee:	Gianluca Petrillo	**% Done:**	100%
Category:	Simulation	**Estimated time:**	4.00 hours
Target version:		**Spent time:**	4.00 hours
Occurs In:	v06_55_00	**Co-Assignees:**	
Experiment:	-		

Description

DUNE Photon Library generation now takes ~5000M memory on the cluster. This makes library generation extremely difficult. This issue has been introduced since LArSoft v06_35_00. The module responsible for the extreme memory usage is SimPhotonCounter.

Associated revisions

Revision 150e365f - 10/31/2017 02:14 PM - Gianluca Petrillo

PhotonLibrary does not allocate reflected photons unless requested.

Photon library now allows to specify whether to store information about
reflected photons or not (and reflected photon timing as well,
independently).
PhotonVisibilityService drives that feature from the existing
configuration knobs.
This solves the immediate resource usage bloat reported in issue #17708.

Revision 305b29a7 - 10/31/2017 02:14 PM - Gianluca Petrillo

PhotonLibrary does not allocate reflected photons unless requested.

Photon library now allows to specify whether to store information about
reflected photons or not (and reflected photon timing as well,
independently).
PhotonVisibilityService drives that feature from the existing
configuration knobs.
This solves the immediate resource usage bloat reported in issue #17708.

History

#1 - 09/14/2017 10:12 AM - Alexander Himmel

Our suspicion is that this is related to the introduction of the backtracker. The right solution, probably, is to refactor the library-making from the other uses of this module, but doing this without a lot of copy-paste will require introducing some shared algorithms which we can be re-used in multiple places.

#2 - 09/18/2017 10:23 AM - Lynn Garren

- Status changed from New to Feedback

Jason, are you able to fix this or are you seeking help?

#3 - 09/20/2017 10:42 AM - Jason Stock

We (Alex Himmel and Myself) are seeking help. I wouldn't mind fixing this myself, but I don't have the hours to assign to it right now (nor does Alex), and it significantly impairs our ability to generate photon libraries for DUNE.

#4 - 10/27/2017 03:56 PM - Gianluca Petrillo

Please provide a job configuration to work with.
Photon library generation might be a good candidate.

#5 - 10/27/2017 04:17 PM - Jason Stock

fcl=/pnfs/dune/scratch/users/jstock/OpticalLibraries/OpticalLib_dune10kt_v2_1x2x6/dune10kt_v2_1x2x6_buildopticallibrary_grid.fcl
script=/pnfs/dune/scratch/users/jstock/OpticalLibraries/OpticalLib_dune10kt_v2_1x2x6/OpticalLibraryBuild_Grid_dune.sh

jobsub_submit -e IFDH_CP_MAXRETRIES=5 -e IFDH_CP_MAXRETRIES=5 -e mrb_top=/dune/app/users/jstock/tmpBuild -e mrb_project=dunetpc -e mrb_version=v06_54_00 -e mrb_quals=e14:prof --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC --OS=SL6 --group=dune -f /pnfs/dune/scratch/users/jstock/OpticalLibraries/OpticalLib_dune10kt_v2_1x2x6/dune10kt_v2_1x2x6_buildopticallibrary_grid.fcl --role=Analysis --memory=6500MB --expected-lifetime=16h -dROOT /pnfs/dune/scratch/users/jstock/OpticalLibraries/OpticalLib_dune10kt_v2_1x2x6/root -dFCL /pnfs/dune/scratch/users/jstock/OpticalLibraries/OpticalLib_dune10kt_v2_1x2x6/fcl -dLOG /pnfs/dune/scratch/users/jstock/OpticalLibraries/OpticalLib_dune10kt_v2_1x2x6/log -N 6000 file:///pnfs/dune/scratch/users/jstock/OpticalLibraries/OpticalLib_dune10kt_v2_1x2x6/OpticalLibraryBuild_Grid_dune.sh 6000 50000 dune10kt_v2_1x2x6_buildopticallibrary_grid.fcl

The dune SubmitCommand.sh script is provided under dunetpc/dune/PhotonPropagation/LibraryBuildTools to make submission to the grid easier, and is the tool used for every observed occurrence of this bug.

I would like to verify that this is still an issue after the BackTracker becomes Lazy. I am working on CI for the various experiments right now, and might even be able to push the backtracker update this weekend (I would like to do so before I leave).

I suspect what happens is, when we run the full library we keep all photons, meaning they make it into the ParticleList.

Since the backtracker has been added to SimPhotonCounter, if the particle list is built during the use of simphoton counter then EVERY photon from the even will be loaded into memory. I haven't investigated this yet, but I suspect that is where the memory usage is coming from.

#6 - 10/27/2017 07:02 PM - Gianluca Petrillo

- Category set to Simulation

- Status changed from Feedback to Work in progress

Seems more complicate than I hoped.
Can we take the shortcut and use any of the configuration files already concocted in /pnfs/dune/scratch/users/jstock/OpticalLibraries/OpticalLib_dune10kt_v2_1x2x6/fcl?
I will have to wait dunetpc v06_55_00 anyway.

#7 - 10/27/2017 07:13 PM - Jason Stock

Yeah. We can observe it interactively. I grabbed one of the fcl files from my last grid run. It is at:
/dune/app/users/jstock/pd_library_gen_575557_3683.fcl
But, as you mention, any fcl file from the scratch space you pointed at will work fine. The scripts we run build a different fcl file for each grid job, and some seem to be more memory intensive that others, but every fcl file made during the library generation does get written out in the output, and they will run interactively.

#8 - 10/30/2017 10:27 AM - Lynn Garren

- Assignee set to Gianluca Petrillo

#9 - 10/30/2017 06:21 PM - Gianluca Petrillo

- Assignee deleted (Gianluca Petrillo)

I believe I can reproduce the issue:

```
MemReport  ---------- Memory  Summary ---[base-10 MB]----
MemReport  VmPeak = 5732.05 VmHWM = 4836.08
```

#10 - 10/31/2017 12:50 PM - Gianluca Petrillo

- Status changed from Work in progress to Feedback

- Assignee set to Gianluca Petrillo

- % Done changed from 0 to 20

- Estimated time set to 4.00 h

- Occurs In v06_55_00 added

The source of memory usage is the photon library. You are simulating 3000000 voxels and 120 channels.
For each of the 360000000 of them, three float values are stored:

1. the direct photon count
2. the reflected photon count
3. the delay of arrival of the reflected photon counts.

Before the introduction of reflections, you would have used only the first item, which is 1.3 GB of data already.
Now you are using thrice as much, that is the 4 GB of memory that is the bulk of the memory used by the job.

This demand of resources does not only happen when building the photon library, but also when using it, as it is the size of the library itself.

The first relevant question is: do you care of the reflected light?
If the answer is no, we can probably manage to have the photon library create the two additional maps for the reflected light only on demand.
If you do care about reflected light, then you have to accept that storing 3M positions for 120 channels does take that amount of space.
In that case, a parametrisation of the map might be the only viable solution.

Incidentally: the whole system of photon library use and creation is horribly designed. A complete revamp would be beneficial; but that will not solve DUNE data size problem.

#11 - 10/31/2017 02:48 PM - Gianluca Petrillo

- Status changed from Feedback to Resolved

- % Done changed from 20 to 100

In the meanwhile, a fix has been pushed into develop branch of larsim as larsim:150e365fa88e3adb249cd42cac4f9230a520d382.
Now phot::PhotonLibrary allocates memory for reflected photons only when required, the default from PhotonVisibilityService being no.
This reduces the memory usage when running without reflected photon simulation from three times 1.3 GB to "just" a single 1.3 GB:

```
MemReport  ---------- Memory  Summary ---[base-10 MB]----
MemReport  VmPeak = 2852.13 VmHWM = 1957.81
```

#12 - 10/31/2017 05:21 PM - Alexander Himmel

Thanks for investigating!

We may want to use reflected light eventually, but for right now we definitely want the ability to run without it to save the memory. Is there something simple that can be done ASAP?

Yes, in the long run the system has issues, and I can't imagine any library method that will work for something the size of the FD. Investigations into replacements are on-going.

#13 - 10/31/2017 05:26 PM - Lynn Garren

This fix will be in the upcoming larsoft release, which is about to be tagged. I hope that is soon enough for you.

#14 - 10/31/2017 05:39 PM - Alexander Himmel

Absolutely! Thanks for the fast turn around.

#15 - 11/01/2017 09:50 AM - Lynn Garren

larsoft v06_55_01

#16 - 11/09/2017 11:38 AM - Gianluca Petrillo

- Status changed from Resolved to Closed