SBND code - Bug #23722

Optical hit finder segfaults when processing SBND MC

12/06/2019 08:55 AM - Dominic Brailsford

Status:	Resolved	**Start date:**	12/06/2019
Priority:	Urgent	**Due date:**	
Assignee:	Iker de Icaza Astiz	**% Done:**	100%
Category:	Reconstruction	**Estimated time:**	0.00 hour
Target version:			
First Occurred:	v08_36_01_3_MCP2_0	**Occurs In:**	v08_36_01_3_MCP2_0

Description

In MCP2.0. we've found that between 10% and 20% of reconstruction jobs fail with a segfault, which is unfortunate as we get very little extra information.
From the local testing done by the production team, it looks like this issue is caused by the optical hit finder.

The MCP2.0 version of sbndcode is v08_36_01_3_MCP2_0 (it was not available in the drop down list...).

I've assigned this to Diego as convenor of the optical group. Please feel free to pass the issue on to the relevant parties.

We've copied a subset of the problematic detsim input files for one of the samples here:
/pnfs/sbnd/persistent/sbndpro/MCP2_0/opHitFinderSegFault/prodsingle_electron_pi+_bnblike_forward_detsim
To help get started, event 84 in
/pnfs/sbnd/persistent/sbndpro/MCP2_0/opHitFinderSegFault/prodsingle_electron_pi+_bnblike_forward_detsim/detsim-33b6d20e-0ea2-4517-80ab-5252c3b73366.root segfaults with optical hit finding enabled.

History

#1 - 12/06/2019 10:23 AM - Diego Gamez

- Assignee changed from Diego Gamez to Iker de Icaza Astiz

#2 - 12/06/2019 10:55 AM - Thomas Brooks

- First Occurred changed from v08_19_00 to v08_36_01_3_MCP2_0

- Occurs In v08_36_01_3_MCP2_0 added

- Occurs In deleted (v08_19_00)

#3 - 12/12/2019 10:59 PM - Iker de Icaza Astiz

- % Done changed from 0 to 20

This bug is caused by an index out of bounds on the function TV1D_denoise, below is the original version of the function. As its author wrote it https://www.gipsa-lab.grenoble-inp.fr/~laurent.condat/publis/Condat-fast_TV-SPL-2013.pdf
Note that the function on Op Hit Finder was a verbatim copy of this.

At the author's website: https://www.gipsa-lab.grenoble-inp.fr/~laurent.condat/ I found another version that uses doubles instead of float, but other than that is identical.

For the specific first file that the production team announced to fail, I've managed to fix the issue by changing the floats to doubles. Some other files also benefited from this.

I'm yet to find out why this happens. I suspect the bug hadn't manifested before through a combination of demoting doubles to floats and just not being able to run that many waveforms without running into other issues first.

My personal opinion is that the code below is really hard to follow, hence I committed an unwrapped equivalent version.

```
void TV1D_denoise(float* input, float* output, const int width, const float lambda) {
    if (width>0) {                /*to avoid invalid memory access to input[0]*/
        int k=0, k0=0;            /*k: current sample location, k0: beginning of current segment*/
        float umin=lambda, umax=-lambda;    /*u is the dual variable*/
        float vmin=input[0]-lambda, vmax=input[0]+lambda;    /*bounds for the segment's value*/
        int kplus=0, kminus=0;     /*last positions where umax=-lambda, umin=lambda, respectively*/
        const float twolambda=2.0*lambda;    /*auxiliary variable*/
        const float minlambda=-lambda;        /*auxiliary variable*/
        for (;;) {                /*simple loop, the exit test is inside*/
```

```
            while (k==width-1) {    /*we use the right boundary condition*/
                if (umin<0.0) {            /*vmin is too high -> negative jump necessary*/
                    do output[k0++]=vmin; while (k0<=kminus);
                    umax=(vmin=input[kminus=k=k0])+(umin=lambda)-vmax;
                } else if (umax>0.0) {    /*vmax is too low -> positive jump necessary*/
                    do output[k0++]=vmax; while (k0<=kplus);
                    umin=(vmax=input[kplus=k=k0])+(umax=minlambda)-vmin;
                } else {
                    vmin+=umin/(k-k0+1);
                    do output[k0++]=vmin; while(k0<=k);
                    return;
                }
            }
            if ((umin+=input[k+1]-vmin)<minlambda) {        /*negative jump necessary*/
                do output[k0++]=vmin; while (k0<=kminus);
                vmax=(vmin=input[kplus=kminus=k=k0])+twolambda;
                umin=lambda; umax=minlambda;
            } else if ((umax+=input[k+1]-vmax)>lambda) {    /*positive jump necessary*/
                do output[k0++]=vmax; while (k0<=kplus);
                vmin=(vmax=input[kplus=kminus=k=k0])-twolambda;
                umin=lambda; umax=minlambda;
            } else {     /*no jump necessary, we continue*/
                k++;
                if (umin>=lambda) {        /*update of vmin*/
                    vmin+=(umin-lambda)/((kminus=k)-k0+1);
                    umin=lambda;
                }
                if (umax<=minlambda) {    /*update of vmax*/
                    vmax+=(umax+lambda)/((kplus=k)-k0+1);
                    umax=minlambda;
                }
            }
        }
    }
}
```

#4 - 12/12/2019 11:45 PM - Iker de Icaza Astiz

- % Done changed from 20 to 60

- File foo.png added

I have looked into:

- changing the do/while loops to only use whiles
- completely unfold the code, thinking that perhaps a compiler optimisation was 'causing an unexpected behaviour
- I implemented a python version of the unfolded code, still presented issues

On Tuesday 10th I got together with Franciole to continue working on this. The first challenge was to find a troublesome waveform, as I noted on my previous post using doubles solved various files.

We've isolated a problematic waveform originating from a cosmic generation of the CI system. Then using the python standalone I made some plots. Without any initial problem. Later I found out that I wasn't using the full precision and sure enough with full double the python script crashed.

Oddly enough double lambda = 10.0 causes the script to crash, any other value close to 10 is ok, for instance 9.999999999999999 or 10.00000000000001. This value was chosen following experimental test runs.

Attached, you can find examples of how the problematic waveform gets denoised with varying values of lambda. On a quick glance, there's nothing odd about this particular waveform.

I've put checks in place so that if the denoiser function goes haywire it stops and tries again with a different lambda value, and up to 5 times.

I ran ophit on all of the files in /pnfs/sbnd/persistent/sbndpro/MCP2_0/opHitFinderSegFault/prodsingle_electron_pi+_bnblike_forward_detsim without any issue, in fact not a single one required a change of lambda. At least for this sample the fix of floats to doubles was enough.

#5 - 12/13/2019 06:19 PM - Iker de Icaza Astiz

Running the CI tests:

```
 test_runner develop_test_sbndcode --parallel-limit=4
The current parallel limit is: 4
Test Suite:  develop_test_sbndcode
Expanded:  nucosmics_detsim_quick_test_sbndcode nucosmics_reco_basic_quick_test_sbndcode single_anatree_quick_
```

```
test_sbndcode nucosmics_gen_quick_test_sbndcode compilation_test_sbndcode single_reco_basic_quick_test_sbndcod
e nucosmics_g4_quick_test_sbndcode nucosmics_anatree_quick_test_sbndcode single_g4_quick_test_sbndcode single_
gen_quick_test_sbndcode single_detsim_quick_test_sbndcode
...Scaled cpu usage 167.255636 not in 200.000000:800.000000 range
W Scaled cpu usage 226.595507 not in 400.000000:1500.000000 range
W.Memory usage 1426616.000000 not in 800000.000000:1400003.000000 range
W..Scaled cpu usage 525.962857 not in 550.000000:1703.000000 range
W W
6 tests passed (54%), 0 tests failed, 5 tests with warnings, 0 tests skipped, out of 11
Not updating any reference files
```

The memory usage went up in nucosmics_anatree_quick_test_sbndcode test and, oddly enough, the CPU usage went down in a few others.

Only one time the TV1D_denoise() failed to run on a first attempt, and it succeeded at the second attempt. No segfaults whatsoever.

The number of data products changed in nucosmics_g4_quick_test_sbndcode, for the most part minor changes only. Biggest differences are:

- < G4 | largeant | | std::vector<simb::MCParticle> | 24747

- > G4 | largeant | | std::vector<simb::MCParticle> | 20371

- < G4 | largeant | | art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo> | 24747

- > G4 | largeant | | art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo> | 20371

```
Compare products sizes for nucosmics_g4_test_sbndcode_Current.root output stream.

Check for differences in the size of data products
difference(s)

14,15c14,15
< G4 | mcreco |  | std::vector<sim::MCShower> | 247
< G4 | largeant |  | std::vector<simb::MCParticle> | 20186
---
> G4 | mcreco |  | std::vector<sim::MCShower> | 248
> G4 | largeant |  | std::vector<simb::MCParticle> | 20197
19c19
< G4 | largeant |  | art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo> | 20186
---
> G4 | largeant |  | art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo> | 20197
21c21
< G4 | largeant |  | std::vector<sim::SimChannel> | 10647
---
> G4 | largeant |  | std::vector<sim::SimChannel> | 10648
36,37c36,37
< G4 | mcreco |  | std::vector<sim::MCShower> | 221
*< G4 | largeant |  | std::vector<simb::MCParticle> | 24747*
---
> G4 | mcreco |  | std::vector<sim::MCShower> | 236
*> G4 | largeant |  | std::vector<simb::MCParticle> | 20371*
39c39
< G4 | mcreco |  | std::vector<sim::MCTrack> | 185
---
> G4 | mcreco |  | std::vector<sim::MCTrack> | 199
41c41
< G4 | largeant |  | art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo> | 24747 *
---
> G4 | largeant |  | art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo> | 20371 *
43c43
< G4 | largeant |  | std::vector<sim::SimChannel> | 10646
---
> G4 | largeant |  | std::vector<sim::SimChannel> | 10652
```

#6 - 12/13/2019 06:25 PM - Iker de Icaza Astiz

- % Done changed from 60 to 100

- Status changed from New to Resolved

I have pushed my code to feature/icaza_ophit, I believe this is now solved.

I'll handle the increased memory usage in the future, but the urgent matter has been taken care of.

Files