dunetpc - Bug #13749

Data prep service causing jobs to exceed memory usage

09/01/2016 07:33 PM - Thomas Warburton

Status:	Closed	**Start date:**	09/01/2016
Priority:	Normal	**Due date:**	
Assignee:	Tingjun Yang	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

When running the reconstruction for the 10 kt detector and the 35 ton CRY milliblock samples the jobs use over 2 GB memory.
Looking at the memory tracker information for the jobs it appears that this is caused by the data prep service.

I attach the file paths to two files which I have used to verify this:
/dune/data2/users/warburton/DataPrep/CorrectedDetsim_countermu_20160902T002056_reco_milliblock.root
/pnfs/dune/scratch/dunepro/v06_04_01/detsim/prodMUSUN_dune10kt/10115587_48/prodMUSUN_dune10kt_28_20160826T220116_gen_c3882a40-e647-4aee-b6c1-d01b55ced06b_g4_detsim.root

In this directory there is also some piped output from me running the jobs where you can see the memory usage from each job.
/dune/data2/users/warburton/DataPrep/

The fcl files which I used were the standard reconstruction fcl files:
standard_reco_dune35tsim_milliblock.fcl
standard_reco_dune10kt.fcl

David, could you take a look at this please?

History

#1 - 09/01/2016 09:23 PM - David Adams

I will have a look but it may not be before next week.

You might try running with only the data prep producer.

#2 - 09/02/2016 10:44 AM - Thomas Warburton

Hi David,

We cannot finish MCC7.1 without fixing this.

I just ran the data prep separately and it appears to see reduced memory usage (for the milliblock sample at least), which is good. However when I then run the event display I am seeing that the pedestals have been incorrectly applied.
Looking at the output of ART_DEBUG_CONFIG for standard_detsim_dune35t.fcl and standard_detsim_dune35t_milliblock there are a number of differences:

- The pedestal for the milliblock sample does not use fixed pedestals, but the data pedestals.
- The AdcSuppress service is different and there is also an extra service called AdcSuppressonService (with typo)
- There are also extra services such as AdcDeconvolutionService

The first two problems go away if I add the fcl overwrites at the bottom of standard_detsim_dune35t.fcl to standard_detsim_dune35t_milliblock.fcl. I think this means that they are getting overwritten when we update the services to be used for the milliblock sample.

When I then run the event display on this file I do not see errors with the pedestals in the raw or reconstructed views.
I do see strange black bars in the reconstructed tab for TPCs 4 and 5....
Is it ok for me to push these changes?

#3 - 09/02/2016 02:13 PM - David Adams

Karl:

For validation of the new 35t data prep, please see #12701. I did check that the new data prep gave the same results as the old caldata but maybe the old milliblock reco also had problems?

I am not sure what you propose to do. The overrides in standard_detsim_dune35t.fcl only affect producers that run after data prep. Here is the full file:

```
#include "standard_reco_dune35tsim.fcl"

physics.producers.dbclusterdc.HitsModuleLabel:          "lineclusterdc"
physics.producers.costrkdc.ClusterModuleLabel:          "lineclusterdc"

physics.producers.dbcluster.HitsModuleLabel:            "hit35tcc"
physics.producers.costrk.ClusterModuleLabel:            "linecluster"

#Pandora configurations
physics.producers.pandoradc.HitFinderModuleLabel:           "dcheatcc"
physics.producers.pandora.HitFinderModuleLabel:             "linecluster"

physics.producers.linecluster.ClusterCrawlerAlg.TimeDelta:    [2,3,3]
physics.producers.lineclusterdc.ClusterCrawlerAlg.TimeDelta:    [2,3,3]
```

Changing these is fine with me but should not affect the output of data prep.

Am I missing something?

da

#4 - 09/02/2016 02:31 PM - Thomas Warburton

Potentially, though I was not aware of any. Regardless pushing this fix will rectify it.

You show the fcl file for standard_reco_dune35tsim_cc.fcl. I was referring to standard_detsim_dune35t.fcl namely the following lines;

1. Use fixed values instead of DB for pedestals.
   services.DetPedestalService: @local::dune_fixedpeds

1. DetSim flags.
   physics.producers.daq.NoiseOn: true
   physics.producers.daq.PedestalOn: true
   physics.producers.daq.DistortOn: false
   physics.producers.daq.SuppressOn: true

1. DetSim services.
   services.SimChannelExtractService: @local::scxgeneric
   services.ChannelNoiseService: @local::chnoiseold
   services.PedestalAdditionService: @local::padprovided
   services.AdcDistortService: @local::stuckbits
   services.AdcSuppressService: @local::zslegacy
   services.AdcCompressService: @local::cmpblock

1. Disable bad channels.
   #services.IChannelStatusService.BadChannels: [ ]

I will push these changes to standard_detsim_dune35t_milliblock.fcl.

However, this is a separate point to data prep exceeding memory usage which still exists, most notably in the far detector geometry where running data prep separately did not reduce memory usage.

#5 - 09/07/2016 07:57 AM - David Adams

Karl:

Sorry to be slow in responding. It seems there may be multiple issues being discussed here. It is easier for me if we open separate tickets in such cases.

Let's first settle the problems with detsim which may or may not be the source of the memory increase that led to this ticket. If I go back one release to v06_04_01, I see this for standard_detsim_dune35t_milliblock.fcl:

```
#include "standard_detsim_dune35t.fcl"

services.DetectorPropertiesService.NumberTimeSamples:  32000
services.DetectorPropertiesService.ReadOutWindowSize:  32000
outputs.out1.fileName: "%ifb_%tc_detsim_milliblock.root"
```

I.e. milliblock uses the same configuration as the standard 35 ton except for number of samples. This should have been validated when I changed the standard 35-ton detsim. I gave a validation talk for the latter at the April 11 meeting of the FD sim/reco group but I don't see any evidence there that I validated milliblock production.

Did you find that this milliblock fcl did not work? Did you report that in Redmine or to me? Evidently you or someone found problems because the fcl was changed in v06_05_00 and is changed again in the head as you describe here. The new fcl duplicates a lot of code that is in standard_detsim_dune35t.fcl. This is probably not desirable as we would typically want future changes made in one to also appear in the other.

Thanks in advance for helping me to understand what is going on here. And if all this has been discussed in another ticket that has slipped my mind, please give me its ID and accept my apologies.

Thanks.
David

#6 - 09/07/2016 01:52 PM - Thomas Warburton

David:

I would prefer one ticket, even if we just go through issues sequentially.

Tingjun and I decided (privately) that the milliblock sample should be changed to 15,000 ticks to better represent the data. This was the goal of the milliblock sample in the first place where we expected events which we 10 drift windows long (at 500 V cm-1).

To achieve this I changed the parameters in services_dune.fcl but noticed that the change in the number of samples was not being passed to the sample correctly. This is because the NumberTimeSamples was hardcoded in this fcl file, so it was not inheriting from dune35t_milliblock_detsim_services. This was the reason for the first change made on 1st September.

When I then ran the detsim stage I noticed that many of the fcl parameters were now not set correctly, and so copied in the overrides from standard_dune35t_detsim.fcl to fix this, as you not above. This was the reason for the second commit made on 2nd September.
Looking at it with fresh eyes, the physics.producers.daq.XXXX: YYY lines are probably not needed as these will not be overwritten.
However the services.XXX lines are required as they are not set correctly in dune35t_milliblock_detsim_services or dune35t_detsim_services hence the need to overwrite them in standard_dune35t_detsim.fcl and here.

The third recent commit also made on 2nd September was because I noticed I was using dune35t_milliblock_reco_services not dune35t_milliblock_detsim_services.

I hope that helps to explain the commits which I made.
Karl

#7 - 09/08/2016 07:06 AM - David Adams

So this should work?:

```
#include "standard_detsim_dune35t.fcl"

services.DetectorPropertiesService.NumberTimeSamples:  15000
services.DetectorPropertiesService.ReadOutWindowSize:  15000
outputs.out1.fileName: "%ifb_%tc_detsim_milliblock.root"
```

Thanks.

da

#8 - 09/08/2016 09:02 AM - Thomas Warburton

David:

Yes, that would work. However that seems even less optimal to me as the user is now not using dune35t_milliblock_detsim_services but is instead using dune35t_detsim_services and so the user has to know every difference between the two services. Luckily in this case the only difference is changing the number of the readout samples but there is no reason to think that this would always be the case.

Is there a good reason why the service overrides cannot go in services_dune.fcl? This is surely the logical place for them to go and not in the standard_detsim_dune35t fcl file.

Karl

#9 - 09/08/2016 09:29 AM - David Adams

Karl:

I agree it is find to have and use dune35t_milliblock_detsim_services. Or to do as I suggest which I think is what we had before the change in the detsim service.

I just think the current top-level fcl is rather convoluted and difficult to understand.

Thanks.

da

#10 - 09/08/2016 10:33 AM - Thomas Warburton

Ok, so we can leave standard_detsim_dune35t_milliblock.fcl as it currently is in the head of dunetpc?

Going back to the memory issue at hand.

- There still remains a problem with memory usage when running dataprep as a standalone process on the full far detector geometry.
- When running data prep by itself on the 35 ton milliblock sample the memory usage is acceptable (at least for a 10 event sample), when running it as part of standard_reco_dune35tsim_milliblock.fcl the memory usage sneaks over 2 GB (for 10 events) so I think in a full sample this would still cause problems.

Karl

#11 - 09/15/2016 07:41 AM - David Adams

- Status changed from New to Assigned

Karl:

The current version (option 1) of standard_detsim_dune35t_milliblock.fcl has this:

```
#include "standard_reco_dune35tsim.fcl"

includedServices: @local::services

services: {
    @table::includedServices
    @table::dune35t_milliblock_reco_services
}

includedServices: @erase

outputs.out1.fileName: "%ifb_%tc_reco_milliblock.root"
```

So in addition to copying the producer chain from standard_reco_dune35tsim, it copies all the services and then overwrites those defined in dune35t_milliblock_reco_services. I suppose this is OK until we decide there is a service in standard_reco_dune35tsim that we would like to remove rather than overwrite.

An alternative (option 2) is:

```
#include "standard_reco_dune35tsim.fcl"

services: {
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "reco_hist.root" }
  TimeTracker:       {}
  MemoryTracker:     {}
  RandomNumberGenerator: {} #ART native random number generator
  message:      @local::dune_message_services_prod_debug
  FileCatalogMetadata:  @local::art_file_catalog_mc
  @table::dune35t_milliblock_reco_services
}

outputs.out1.fileName: "%ifb_%tc_reco_milliblock.root"
```

We no longer inherit any services from standard_reco_dune35tsim but I have verified this is equivalent to the current version. I prefer this because if someone adds something to standard reco it is not automatically added to milliblock reco. If they want it in both, they would add it to the shared block in services_dune.

Perhaps even better (option 3) would be to add include the explicit services above

```
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "reco_hist.root" }
  TimeTracker:       {}
  MemoryTracker:     {}
  RandomNumberGenerator: {} #ART native random number generator
  message:      @local::dune_message_services_prod_debug
  FileCatalogMetadata:  @local::art_file_catalog_mc
```

in dune35tsim_reco_services and dune35t_milliblock_reco_services (e.g. by including them in shared definition). Then standard_detsim_dune35t_milliblock.fcl could simply be

```
#include "standard_reco_dune35tsim.fcl"

services: @local::dune35t_milliblock_reco_services
```

Karl and Tingjun, please let me know which of these options you prefer. In the meantime, I will start to look at the memory issue.

#12 - 09/15/2016 11:42 AM - David Adams

- Assignee changed from David Adams to Thomas Warburton

I processed 10 cosmic events with full milliblock reco (standard_reco_dune35tsim_milliblock.fcl) and see this memory usage:

```
Events with large Vsize (Mbytes)        Vsize      Δ Vsize      RSS          Δ RSS
=============================================================================================
  run: 10000002 subRun: 0 event: 4    1793.836        0        941.609        0
  run: 10000002 subRun: 0 event: 5    1793.836        0        941.629      0.020
  run: 10000002 subRun: 0 event: 6    1793.836        0        941.629        0
  run: 10000002 subRun: 0 event: 7    1793.836        0        941.637      0.008
```

I then split the processing to first run only dataprep:

```
Events with large Vsize (Mbytes)        Vsize      Δ Vsize      RSS          Δ RSS
=============================================================================================
  run: 10000002 subRun: 0 event: 5    1483.090     181.328     564.277     117.523
  run: 10000002 subRun: 0 event: 1    1438.273     348.156     522.445     285.848
  run: 10000002 subRun: 0 event: 2    1301.762        0        445.070      0.449
  run: 10000002 subRun: 0 event: 4    1301.762        0        446.754        0
```

and then the rest of the reco chain (no dataprep):

```
Events with large Vsize (Mbytes)        Vsize      Δ Vsize      RSS          Δ RSS
=============================================================================================
  run: 10000002 subRun: 0 event: 3    1778.629     125.422     925.395     124.750
  run: 10000002 subRun: 0 event: 5    1774.633     30.598      922.152     30.613
  run: 10000002 subRun: 0 event: 8    1765.668     71.762      913.203     71.773
  run: 10000002 subRun: 0 event: 4    1744.035     -35.078     891.539     -34.785
```

Comparing these, I conclude the dataprep adds very little to the memory usage.

Karl, you report we are using more memory than previous production/releases. I suppose this is due to one (or more) of the following:
1. More reco producers are being run
2. One of the reco producers has been modified and uses more memory
3. The upstream generation or simulation is allowing more data through

Of these, only the last can be due to the new dataprep (or detsim). Do you see that the detsim event size is larger now than it was in earlier releases?

For point 1, has the list of producers changed? If so, can you try running with the old list and see how that affects the memory usage?

I will wait for feedback before doing more here.

Thank you.

#13 - 09/15/2016 11:49 AM - Thomas Warburton

- Assignee changed from Thomas Warburton to David Adams

Hi David,

I will look at this further next week. I think that the milliblock sample may now be ok. However I believe that a problem still exists with the far detector geometry. Could you please have a look at this?

I noted in comment #4
However, this is a separate point to data prep exceeding memory usage which still exists, most notably in the far detector geometry where running data prep separately did not reduce memory usage.

Cheers,
Karl

#14 - 09/15/2016 12:05 PM - David Adams

- Assignee changed from David Adams to Thomas Warburton

Karl:

Is this only in full FD or in the 1x2x6 subdetector. Is the problem evident with single muons?

I will repeat the above study with this.

#15 - 09/15/2016 12:18 PM - Thomas Warburton

I did not notice a problem when submitting the 1x2x6 geometries. It was with the MUSUN sample which is simulated in the full FD. The MUSUN sample simulates single muons.

You can find a g4 file to test at this location:
/pnfs/dune/scratch/dunepro/v06_04_01/g4/prodMUSUN_dune10kt/10109056_62/prodMUSUN_dune10kt_70_20160826T220124_gen_87e8d4ca-20f6-420f-b959-b366c3f8005d_g4.root
You can find a detsim file to test at this location:
/pnfs/dune/scratch/dunepro/v06_04_01/detsim/prodMUSUN_dune10kt/10115587_8/prodMUSUN_dune10kt_70_20160826T220124_gen_87e8d4ca-20f6-420f-b959-b366c3f8005d_g4_detsim.root

Cheers.

#16 - 09/15/2016 01:06 PM - David Adams

Can you point me to the gen, g4, detsim and reco fcl files used for the sample? Thanks.

#17 - 09/15/2016 01:14 PM - David Adams

For FD1x2x6 with single muons (not yet what Karl requested), I do see that dataprep affects the total memory used.

Full reco:

```
Events with large Vsize (Mbytes)         Vsize      Δ Vsize      RSS          Δ RSS
================================================================================
  run: 20000014 subRun: 0 event: 8    1649.172     240.316     825.141     237.184
  run: 20000014 subRun: 0 event: 6    1569.898     250.289     745.324     246.637
  run: 20000014 subRun: 0 event: 4    1521.809     230.266     700.777     230.219
  run: 20000014 subRun: 0 event: 3    1471.781     180.238     647.152     176.617
```

Dataprep only:

```
Events with large Vsize (Mbytes)         Vsize      Δ Vsize      RSS          Δ RSS
================================================================================
  run: 20000014 subRun: 0 event: 4    1459.555     190.254     630.758     187.145
  run: 20000014 subRun: 0 event: 8    1449.551     180.250     621.941     178.328
  run: 20000014 subRun: 0 event: 7    1449.531     180.230     621.543     177.930
  run: 20000014 subRun: 0 event: 3    1414.477     145.176     585.656     142.043
```

All but dataprep:

```
Events with large Vsize (Mbytes)          Vsize      Δ Vsize      RSS          Δ RSS
================================================================================
  run: 20000014 subRun: 0 event: 7     1347.062     -44.836     525.684     -43.113
  run: 20000014 subRun: 0 event: 10    1306.699     27.953      485.375     27.984
  run: 20000014 subRun: 0 event: 8     1279.906     -67.156     458.551     -67.133
  run: 20000014 subRun: 0 event: 9     1278.746     -0.957      457.391     -0.957
```

#18 - 09/16/2016 10:51 AM - David Adams

It would be helpful to have a memory report for every event rather than those with largest memory and Delta-memory (which is anyway incorrect--see #13873). To do this, add "--memcheck-db mem.dat" to the lar command line and then read the mysql file with

```
sqlite3 -header -column mem.dat 'select * from EventInfo;'
```

Thanks to Kyle Knoepfel for this information.

#19 - 09/16/2016 11:25 AM - David Adams

Here are some all-event memory reports, again for FD 1x2x6.

For full reco:

```
Run         Subrun      Event       Vsize       DeltaVsize  RSS         DeltaRSS
----------  ----------  ----------  ----------  ----------  ----------  ----------
20000014    0           1           1426.453125  344.453125  597.64453125  339.47265625
20000014    0           2           1446.71875   155.253906  621.04296875  152.83984375
```

20000014	0	3	1471.757812	180.238281	644.421875	176.03515625
20000014	0	4	1521.777343	230.257812	698.2265625	229.8203125
20000014	0	5	1291.519531	0	468.47265625	0
20000014	0	6	1566.824218	275.304687	742.4609375	273.98828125
20000014	0	7	1665.492187	260.359375	838.765625	256.66015625
20000014	0	8	1645.46875	240.335937	819.62109375	237.5078125
20000014	0	9	1405.132812	0	582.13671875	0.00390625
20000014	0	10	1405.132812	0	582.1640625	0.02734375

For DataPrep only:

Run	Subrun	Event	Vsize	DeltaVsize	RSS	DeltaRSS
20000014	0	1	1401.609375	341.96875	573.2109375	335.375
20000014	0	2	1409.292968	140.175781	582.125	138.769531
20000014	0	3	1414.308593	145.191406	585.2148437	141.859375
20000014	0	4	1459.371093	190.253906	629.7421875	186.386718
20000014	0	5	1269.117187	0	443.3554687	0
20000014	0	6	1529.421875	260.304687	701.703125	258.347656
20000014	0	7	1444.320312	175.203125	618.0390625	174.683593
20000014	0	8	1449.335937	180.21875	619.3554687	176
20000014	0	9	1269.117187	0	443.3554687	0
20000014	0	10	1404.304687	135.1875	576.3007812	132.945312

For all but DataPrep:

Run	Subrun	Event	Vsize	DeltaVsize	RSS	DeltaRSS
20000014	0	1	1166.8671875	14.1640625	342.0234375	21.98046875
20000014	0	2	1198.0898437	21.4882812	374.5625	25.66796875
20000014	0	3	1218.2773437	20.1875	394.7617187	20.19921875
20000014	0	4	1218.265625	-0.0117187	394.8164062	0.0546875
20000014	0	5	1218.265625	0	394.8164062	0
20000014	0	6	1387.8671875	169.601562	563.78125	168.9648437
20000014	0	7	1347.6445312	-44.417968	524.2226562	-42.6640625
20000014	0	8	1280.6601562	-66.859375	457.2578125	-66.8398437
20000014	0	9	1280.578125	0	457.1796875	0.00390625
20000014	0	10	1307.4296875	26.8515625	484.0585937	26.87890625

And doing nothing (no producers):

Run	Subrun	Event	Vsize	DeltaVsize	RSS	DeltaRSS
20000014	0	1	1055.3671875	0	235.51171875	0.0234375
20000014	0	2	1126.6054687	0	300.765625	0
20000014	0	3	1126.6054687	0	300.765625	0
20000014	0	4	1126.6054687	0	302.4296875	0
20000014	0	5	1126.6054687	0	302.4296875	0
20000014	0	6	1131.609375	0	304.16796875	0
20000014	0	7	1129.4179687	0	304.16796875	0
20000014	0	8	1129.3867187	0	304.16796875	0
20000014	0	9	1129.3867187	0	304.16796875	0
20000014	0	10	1129.3867187	0	304.16796875	0

Kyle tells me the Vsize number is after event processing: "The Vpeak and RSS values correspond to the current memory usage, captured and recorded at the after-event boundary.".
Presumably Vsize-DeltaVsize is the memory usage before the event is processed.

#20 - 09/16/2016 11:44 AM - David Adams

I did find a small memory leak in DataPrep. In the case now ROIs were found in a channel I was creating a Wire but not putting it in the event store where it would be deleted at the end of event processing. This will soon go into dunetpc develop.

The above memory reports (taken after fixing that) show no evidence of a large memory leak. It should be possible to reduce the memory usage by processing one APA or one APA plane at a time. I will have to check if the geometry supports this yet.

#21 - 09/16/2016 11:47 AM - David Adams

Karl:

The above fix is in the current version (4f57b804e75ff0daef0839ffbde26e231965b001) of dunetpc. Can you check if this resolves your problem?

#22 - 09/16/2016 11:57 AM - David Adams

Karl and Tingjun:

Can you tell me which for which release the memory consumption was significantly less? I would like to generate and compare with the above values.

Is or was someone monitoring this for every release?

#23 - 09/19/2016 07:35 AM - David Adams

Issue #13877 proposes a way to reduce dataprep memory consumption for detectors with many APAs. It may provide the resolution for this issue.

#24 - 09/22/2016 03:51 PM - Tingjun Yang

I observe large memory usage when I try to run the reconstruction fcl file on data and save intermediate product caldata:noiseRemoved. It uses more than 8 G memory so all grid jobs failed. Here is the fcl file I used:
/dune/data/users/tjyang/lbne35t/standard_reco_dune35tdata.fcl

I will try to see if there is a memory leak using valgrind.

#25 - 09/22/2016 04:23 PM - Tingjun Yang

Additional information.

When setting services.RawDigitPrepService.DoIntermediateStates: false
MemoryTracker General SUMMARY (all numbers in units of Mbytes)

```
Peak virtual memory usage (VmPeak)  : 1683.73 Mbytes
  Peak resident set size usage (VmHWM): 805.676 Mbytes
```

Modules with large Vsize (Mbytes)	Vsize	Δ Vsize	RSS	Δ RSS
reco:caldata:DataPrepModule				
run: 16256 subRun: 1 event: 1	1543.211	428.730	663.070	363.078
run: 16256 subRun: 1 event: 7	1524.309	0.109	712.145	0.109
run: 16256 subRun: 1 event: 5	1524.199	0.656	712.027	0.656
run: 16256 subRun: 1 event: 6	1524.199	0	712.035	0

When setting services.RawDigitPrepService.DoIntermediateStates: true
physics.producers.caldata.IntermediateStates: ["noiseRemoved"]

MemoryTracker General SUMMARY (all numbers in units of Mbytes)

```
Peak virtual memory usage (VmPeak)  : 2681.46 Mbytes
  Peak resident set size usage (VmHWM): 1809.75 Mbytes
```

Modules with large Vsize (Mbytes)	Vsize	Δ Vsize	RSS	Δ RSS
reco:caldata:DataPrepModule				
run: 16256 subRun: 1 event: 9	2599.793	231.289	1721.211	165.656
run: 16256 subRun: 1 event: 8	2518.285	229.879	1639.641	164.242
run: 16256 subRun: 1 event: 7	2437.480	232.137	1558.723	166.352
run: 16256 subRun: 1 event: 6	2356.418	241.109	1477.660	175.152

There does seem to be a memory leak in DataPrep when we save intermediate product.

Tingjun

Tingjun Yang wrote:

> I observe large memory usage when I try to run the reconstruction fcl file on data and save intermediate product caldata:noiseRemoved. It uses more than 8 G memory so all grid jobs failed. Here is the fcl file I used:
> /dune/data/users/tjyang/lbne35t/standard_reco_dune35tdata.fcl
>
> I will try to see if there is a memory leak using valgrind.

#26 - 09/23/2016 09:13 AM - David Adams

- Status changed from Assigned to Feedback

- Assignee changed from Thomas Warburton to Tingjun Yang

Tingjun:

As discussed above, the log report of "large" memory usage is buggy. Could you follow the instructions for comment 18 above (#13749#note-18) and provide the report for all (and at least 20) events? Thanks.

#27 - 10/03/2016 04:32 AM - Thomas Warburton

Hi David,

Apologies for not replying for such a long time, I got overwhelmed with moving...At the end of last week and this morning I tried re-running dataprep on the 35 ton CRY and Far Detector geometries.

The good news is that I now definitely agree that the 35 ton geometry is fine. I am going to push through the last of the milliblock MCC samples now... However for the Far Detector I am still seeing a problem even after the fix. I have been using v06_07_00 so that should have your fix applied in it.

I ran standard_detsim_dune10kt.fcl on this file /pnfs/dune/scratch/dunepro/v06_04_01/g4/prodMUSUN_dune10kt/10109056_90/prodMUSUN_dune10kt_52_20160826T220138_gen_d78e3ff0-bbcf-4208-b9bd-ffe46b21a2dc_g4.root to make sure that I had the latest version of detsim just in case that was somehow affecting things.

I then ran the following fcl files on the output of that file (first twenty events).
standard_reco_dune10kt.fcl

Run	SubRun	Event	Vsize	DeltaVsize	RSS	DeltaRSS
20000031	53	5201	2730.625	1594.7265625	1962.18359375	1591.984375
20000031	53	5202	4427.80078	1697.171875	3661.96875	1699.261718
20000031	53	5203	2724.26562	0	1958.62890625	0.0078125
20000031	53	5204	2724.26562	0	1958.6328125	0.00390625
20000031	53	5205	2724.26562	0	1958.63671875	0.00390625
20000031	53	5206	2724.26562	0	1958.640625	0.00390625
20000031	53	5207	2724.26562	0	1958.6484375	0.0078125
20000031	53	5208	4426.26562	1702	3660.51953125	1701.871093
20000031	53	5209	2724.28906	0	1958.734375	0.00390625
20000031	53	5210	2724.28906	0	1958.73828125	0.00390625
20000031	53	5211	2724.28906	0	1958.74609375	0.0078125
20000031	53	5212	2724.28906	0	1958.75	0.00390625
20000031	53	5213	2724.28906	0	1958.75390625	0.00390625
20000031	53	5214	2724.28906	0	1958.7578125	0.00390625
20000031	53	5215	4460.93359	1736.6445312	3695.25390625	1736.496093
20000031	53	5216	4468.16406	1743.875	3702.48828125	1743.722656
20000031	53	5217	2724.28906	0	1958.7890625	0.0078125
20000031	53	5218	4499.22656	1774.9375	3734.48828125	1775.699218
20000031	53	5219	4469.86718	1745.578125	3706.734375	1747.050781
20000031	53	5220	4330.15625	1605.8671875	3567.125	1605.800781

standard_reco_dune10kt.fcl (Just data prep)

Run	SubRun	Event	Vsize	DeltaVsize	RSS	DeltaRSS
20000031	53	5201	2544.6328125	1594.73828125	1935.2109375	1591.640625
20000031	53	5202	4235.5234375	1690.88671875	3626.3125	1690.785156
20000031	53	5203	2538.0507812	0	1929.625	0.59375
20000031	53	5204	2538.0507812	0	1930.4453125	0.8203125
20000031	53	5205	2545.8398437	7.7890625	1938.9140625	8.46484375
20000031	53	5206	2538.0507812	-7.7890625	1931.1289062	-7.78515625
20000031	53	5207	2538.0507812	0	1931.2421875	0.11328125
20000031	53	5208	4228.3320312	1690.28125	3621.3359375	1690.09375
20000031	53	5209	2538.0507812	0	1931.2421875	0
20000031	53	5210	2538.0507812	0	1931.2421875	0
20000031	53	5211	2538.0507812	0	1931.2421875	0
20000031	53	5212	2538.0507812	0	1931.2421875	0
20000031	53	5213	2543.8242187	5.7734375	1937.015625	5.7734375
20000031	53	5214	2538.0507812	-5.7734375	1931.2421875	-5.7734375
20000031	53	5215	4260.2929687	1722.2421875	3653.328125	1722.085937
20000031	53	5216	4264.3007812	1726.25	3657.3203125	1726.078125
20000031	53	5217	2538.0507812	0	1931.2421875	0
20000031	53	5218	4295.0898437	1757.0390625	3688.1445312	1756.902343
20000031	53	5219	4263.6171875	1725.56640625	3656.6445312	1725.402343
20000031	53	5220	4123.515625	1585.46484375	3516.6328125	1585.390625

standard_reco_dune10kt.fcl (All but data prep)

Run	SubRun	Event	Vsize	DeltaVsize	RSS	DeltaRSS
20000031	53	5201	1199.015625	0	435.05859375	0.91796875
20000031	53	5202	1248.445312	37.5585937	488.03125	40.7851562
20000031	53	5203	1247.875	0	487.66796875	0.00390625
20000031	53	5204	1247.875	0	487.671875	0.00390625
20000031	53	5205	1247.875	0	487.6796875	0.0078125
20000031	53	5206	1247.875	0	487.68359375	0.00390625
20000031	53	5207	1247.875	0	487.69140625	0.00390625
20000031	53	5208	1264.308593	16.4335937	506.640625	18.9492187
20000031	53	5209	1277.4375	0	519.93359375	0
20000031	53	5210	1277.4375	0	519.93359375	0
20000031	53	5211	1277.4375	0	519.93359375	0
20000031	53	5212	1277.4375	0	519.93359375	0
20000031	53	5213	1277.4375	0	519.93359375	0
20000031	53	5214	1277.4375	0	519.93359375	0
20000031	53	5215	1271.476562	-5.9609375	514.546875	-5.3867187
20000031	53	5216	1271.476562	-1.484375	514.5625	-1.3554687
20000031	53	5217	1271.714843	0	514.80078125	0
20000031	53	5218	1273.664062	1.94921875	516.15625	1.35546875
20000031	53	5219	1275.976562	0	518.625	0
20000031	53	5220	1269.640625	-6.3359375	512.7265625	-5.8984375

Whilst doing this I noted that the showering modules were using a lot of memory so I removed them to isolate any increase which data prep may be causing.

As can be seen from this data prep is affecting the memory usage still, perhaps this brings into focus the need for Issue 13877 to be resolved. Looking through the event display, the events which have large memory usage for data prep are the ones where hits are stored.

The last time which I pushed reconstruction all the way through was for the last MCC challenge which used LArSoft version v05_11_00.

Cheers,
Karl

#28 - 11/16/2016 12:13 PM - David Adams

I have just pushed to dunetpc develop a modification so that data prep now processes each APA separately. This should significantly reduce memory usage in the data prep step where there are many APAs.

#29 - 11/24/2016 07:11 AM - Thomas Warburton

- Status changed from Feedback to Resolved

Hi David,

I just tried running the reconstruction over a newly generated sample of MUSUN files in the far detector (using version v06_15_01). I forgot to run the memset command, but the simple memory usage was much much better - below. This was done for the whole reconstruction chain (taking out pandora, showering and some of the later things for PMTrack as I got some startup errors for them), and as you can see the memory usage is well below 2 GB.

Events increasing Vsize (Mbytes)	Vsize	Δ Vsize	RSS	Δ RSS
run: 20000031 subRun: 0 event: 1	1420.867	249.547	622.152	247.395
run: 20000031 subRun: 0 event: 12	1431.523	7.750	638.926	7.746
run: 20000031 subRun: 0 event: 7	1430.289	6.516	635.914	7.258
run: 20000031 subRun: 0 event: 4	1423.840	2.449	627.309	2.516
run: 20000031 subRun: 0 event: 2	1421.492	0.621	624.871	2.090

Events with large Vsize (Mbytes)	Vsize	Δ Vsize	RSS	Δ RSS
run: 20000031 subRun: 0 event: 12	1431.523	7.750	638.926	7.746
run: 20000031 subRun: 0 event: 13	1430.949	-0.574	638.352	-0.574
run: 20000031 subRun: 0 event: 7	1430.289	6.516	635.914	7.258
run: 20000031 subRun: 0 event: 4	1423.840	2.449	627.309	2.516
run: 20000031 subRun: 0 event: 5	1423.773	-0.066	628.031	0.723

Thankyou!

I think that this means for now at least we can consider the ticket resolved.

#30 - 11/24/2016 08:25 AM - David Adams

- Status changed from Resolved to Closed

Thanks. I close the ticket