dunetpc - Bug #22249

Task # 22198 (New): Address various issues in protodune-sp reconstruction

Possible memory leaks in AdcPedestalFitter_tool.cc

03/29/2019 06:30 PM - Tingjun Yang

Status:	New	**Start date:**	03/29/2019
Priority:	Normal	**Due date:**	
Assignee:	David Adams	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Valgrind showed the following indication of memory leak in AdcPedestalFitter_tool.cc

```
==22281== 61,384 bytes in 15,346 blocks are definitely lost in loss record 62,966 of 63,444
==22281==    at 0x4C2A243: operator new(unsigned long) (vg_replace_malloc.c:334)
==22281==    by 0xD6DE25E: TFormula::HandleParamRanges(TString&) (TFormula.cxx:1176)
==22281==    by 0xD6E093C: TFormula::PreProcessFormula(TString&) (TFormula.cxx:1644)
==22281==    by 0xD6D99C1: TFormula::TFormula(char const*, char const*, bool, bool) (TFormula.cxx:
403)
==22281==    by 0xD6A6354: TF1::TF1(char const*, char const*, double, double, TF1::EAddToList, boo
l) (TF1.cxx:566)
==22281==    by 0x5A16813A: AdcPedestalFitter::getPedestal(AdcChannelData const&) const (AdcPedest
alFitter_tool.cc:321)
==22281==    by 0x5A16686B: AdcPedestalFitter::updateMap(std::map<unsigned int, AdcChannelData, st
d::less<unsigned int>, std::allocator<std::pair<unsigned int const, AdcChannelData> > >&) const (A
dcPedestalFitter_tool.cc:179)
==22281==    by 0x2638B40E: ToolBasedRawDigitPrepService::prepare(std::map<unsigned int, AdcChanne
lData, std::less<unsigned int>, std::allocator<std::pair<unsigned int const, AdcChannelData> > >&,
 std::vector<recob::Wire, std::allocator<recob::Wire> >*, WiredAdcChannelDataMap*) const (ToolBase
dRawDigitPrepService_service.cc:67)
==22281==    by 0x2C139B21: DataPrepModule::produce(art::Event&) (DataPrepModule_module.cc:540)
==22281==    by 0x66C0BE0: art::EDProducer::produceWithFrame(art::Event&, art::ProcessingFrame con
st&) (EDProducer.cc:91)
==22281==    by 0x6797845: art::detail::Producer::doEvent(art::EventPrincipal&, art::ModuleContext
 const&, std::atomic<unsigned long>&, std::atomic<unsigned long>&, std::atomic<unsigned long>&) (P
roducer.cc:125)
==22281==    by 0x275B7D37: art::WorkerT<art::EDProducer>::implDoProcess(art::EventPrincipal&, art
::ModuleContext const&) (WorkerT.h:198)
```

History

#1 - 04/07/2019 04:21 PM - Tingjun Yang

https://cdcvs.fnal.gov/redmine/projects/dunetpc/repository/revisions/9c58f941149a5227276cb93329ecef4d90c71522/entry/dune/DataPrep/Tool/AdcPedestalFitter_tool.cc#L321

#2 - 04/17/2019 01:30 PM - Paul Russo

```
1173 void TFormula::HandleParamRanges(TString &formula)
1174 {
1175    TRegexp rangePattern("\\[[0-9]+\\.\\.[0-9]+\\]");
1176    Ssiz_t *len = new Ssiz_t();
1177    int matchIdx = 0;
1178    while ((matchIdx = rangePattern.Index(formula, len, matchIdx)) != -1) {
1179       int startIdx = matchIdx + 1;
1180       int endIdx = formula.Index("..", startIdx) + 2; // +2 for ".."
1181       int startCnt = TString(formula(startIdx, formula.Length())).Atoi();
1182       int endCnt = TString(formula(endIdx, formula.Length())).Atoi();
1183
1184       if (endCnt <= startCnt)
1185          Error("HandleParamRanges", "End parameter (%d) <= start parameter (%d) in parameter range", endC
```

```
nt, startCnt);
1186
1187        TString newString = "[";
1188        for (int cnt = startCnt; cnt < endCnt; cnt++)
1189           newString += TString::Format("%d],[", cnt);
1190        newString += TString::Format("%d]", endCnt);
1191
1192        // std::cout << "newString generated by HandleParamRanges is " << newString << std::endl;
1193        formula.Replace(matchIdx, formula.Index("]", matchIdx) + 1 - matchIdx, newString);
1194
1195        matchIdx += newString.Length();
1196     }
1197
1198     // std::cout << "final formula is now " << formula << std::endl;
1199 }
```

So we can see here that the memory allocated with "new Ssiz_t()" and assigned to the pointer variable "len" on line 1176 is never released. This is a ROOT bug and should be reported to the ROOT team following these instructions:

https://root.cern.ch/how/report-bug-jira

I have verified that this bug still exists in the most recently released version of ROOT.

#3 - 04/17/2019 01:35 PM - David Adams

Thanks for chasing this down. I have long suspected a leak in TF1. --da

#4 - 04/18/2019 09:09 AM - Tingjun Yang

Bug reported to ROOT team:
https://sft.its.cern.ch/jira/browse/ROOT-10089