art - Support #9333

FHICL Information from Previous Module

06/30/2015 11:45 AM - Martin Frank

Status:	Closed	**Start date:**	06/30/2015
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:	Metadata	**Estimated time:**	0.00 hour
Target version:		**Spent time:**	2.00 hours
Scope:	Internal	**SSI Package:**	fhicl-cpp
Experiment:	NOvA		

Description

I am trying to extract the FHICL information from an upstream module. This works without problems in my analyze method:

```
  // DDT Trigger Decisions
  art::Handle<std::vector<novaddt::TriggerDecision> > td_30, td_20, td_10;
  e.getByLabel("smmgap30", td_30);
  e.getByLabel("smmgap20", td_20);
  e.getByLabel("smmgap10", td_10);

  // Trigger Parameter Set:
  fhicl::ParameterSet p_smt;
  auto IDs = td_20.provenance()->psetIDs();
  if (IDs.size() == 1)
  {
    p_smt = fhicl::ParameterSetRegistry::get(*(IDs.begin()));
  } else {
    // ugly assert that I am too ashamed to print here
  }

  // Run the Trigger Offline
  novaddt::smt::Trigger smt(p_smt);
```

Since this information stays the same event by event, it would be nice to only access it once. Kyle recommended the following incantation that I am currently running in beginRun (but could go into beginJob as well):

```
  for (auto const& reg_pair : art::ProcessHistoryRegistry::get())
  {
    for (auto const& processConfig : reg_pair.second)
    {
      auto const& id = processConfig.parameterSetID();
      auto const& pset = fhicl::ParameterSetRegistry::get(id);
      std::cout << pset.to_indented_string() << std::endl;
    }
  }
```

When Kyle and I looked at the output earlier, we only see the FHICL information from the input file, but not from the currently running module. I have attached the FHICL file together with the output log file. Is there a way to get my hands on the FHICL information from a previous module in beginJob?

History

#1 - 06/30/2015 11:50 AM - Christopher Backhouse

Why do you need to do this? Wouldn't it be easier to just pass the two modules the same configuration in the fcl file? Or for the upstream module to store a product containing the relevant parts of its configuration?

#2 - 06/30/2015 11:54 AM - Rob Kutschke

This is a little off topic but it's worth asking.

Is an assert really what you want to use here? Unless the NOvA build system does something differently than Mu2e's (and art's), the assert is only present if you are compiling with debugging enabled. It is compiled away to nothing in non-debug builds. There are times that this is the behaviour that you want - is that true in this case?

#3 - 06/30/2015 11:55 AM - Christopher Backhouse

We do in fact retain fatal asserts in non-debug builds.

#4 - 06/30/2015 12:02 PM - Kyle Knoepfel

I asked Martin to file this bug report because a call to art::Run::processHistory() or art::ProcessHistoryRegistry::get() is not returning what I would expect given his configuration--that is the salient point here.

#5 - 07/13/2015 03:20 PM - Christopher Green

I quote from the art wiki page AssertsVsExceptions:

> The number one reason for excluding asserts from production code is that every experiment consulted when putting together requirements for art stated that it was a priority to avoid data loss wherever possible. Given that, the developer cannot be the one to decide which errors are fatal and cause immediate termination.

If an assert fires in production code, the output data file (and likely, any histogram file) will be completely unreadable, regardless of how many hours of successful data taking, processing or analysis have been spent up to that point. Even if you are *absolutely sure* that the code at issue will never be run in a data-taking or data processing context, there is still the matter of trying to avoid wasting (*e.g.*, grid) resources.

However, this is all being by way of an aside to the current issue: a ProcessHistory for the current process is not created if no new products are produced for persistence in an event, subrun or run.

If the issue here is making sure that the configuration of an analysis module is consistent in some way with that of a different module (a producer, say), a better way to ensure that might be to use preamble in a FHiCL PROLOG and refer to it in both places with @local::, @table:: or @sequence::.

In the wider context, the information *could* change event by event if the input file were a skim from sources of compatible but slightly different, "shape." You might be better served caching the information in the module and only retrieving the configuration from the registry if the parameter set ID from the provenance changes.

#6 - 07/13/2015 03:34 PM - Christopher Backhouse

Also off-topic:

> If an assert fires in production code, the output data file (and likely, any histogram file) will be completely unreadable, regardless of how many hours of successful data taking, processing or analysis have been spent up to that point. Even if you are absolutely sure that the code at issue will never be run in a data-taking or data processing context, there is still the matter of trying to avoid wasting (e.g., grid) resources.

Except, as we all know, human time is far more valuable than machine time. A clear error and a non-existent output file immediately a problem occurs is far less likely to waste human time than partial, slightly screwed-up, outputs with no sign of trouble unless one mines the logs. I can't even count how many man-hours NOvA has wasted on problems of the latter kind.

I accept that things might be different for live data-taking (though I suspect not).

#7 - 07/13/2015 05:36 PM - Rob Kutschke

Speaking for Mu2e. Our policy is always to throw exceptions in production code and never to use asserts. For example, if you write out a message and soon after assert, the message may trapped in an IO buffer and never be flushed to the log file. In such a case you have lost an important clue to the origin of the problem.

It's true that files that are only partially processed, and not recognized as such, cause problems. Our workflow scripts check the return value of the art executable and "file" the results of bad jobs differently from those of successful jobs. For example the output of failed jobs sits in dCache and never makes it to SAM.

Our grid scripts run with --rethrow-all . So all exceptions cause art to attempt a graceful shutdown. We do not retain any "skip to next event" behaviours

In the end, this is really an issue of what your build system does, not an art issue.

#8 - 07/20/2015 11:23 AM - Christopher Green

- Tracker changed from Bug to Support

- Category set to Metadata

- Status changed from New to Resolved

- % Done changed from 0 to 100

- SSI Package fhicl-cpp added

- SSI Package deleted ()

As notified offline by Martin, the solution for his particular problem was to handle things with FHiCL PROLOG and references such as @local.

#9 - 08/14/2015 07:58 AM - Kyle Knoepfel

- Status changed from Resolved to Closed

Files

monojob.fcl	2.14 KB	06/30/2015	Martin Frank
mono.log	8.18 KB	06/30/2015	Martin Frank