art - Feature #8018

Mark a data product to be dropped

03/04/2015 02:31 PM - Gianluca Petrillo

Status:	Closed	**Start date:**	03/04/2015
Priority:	Normal	**Due date:**	08/01/2015
Assignee:	Kyle Knoepfel	**% Done:**	100%
Category:	I/O	**Estimated time:**	24.00 hours
Target version:	2.10.00	**Spent time:**	8.00 hours
Scope:	Internal	**SSI Package:**	art
Experiment:	LArSoft		

Description

We have a reconstruction producer that produces two collections: "Clusters" and "Hits".
Internally, it's organized into two algorithms, one creating a first tentative list of hits, another starting from those to create the final Clusters and the final Hits.

We now want to split that producer into two modules, so that additional intermediate steps may happen between them. The two new modules have to comminucate though the event and data products: we have to promote the temporary hit to a data product. But we know that those temporary hits are specific to this reconstruction and do not have any useful information once it's finished, therefore we want them to disappear as soon as possible once they are not needed any more.

I can imagine two possible solutions. The ideal one is to have the last module remove the temporary product from the event. This might be against all the rules and models art follows, not to mention it's hard to formalize a proper behaviour.
The other one is to have that temporary product in the event, but not serialized to disk.

This latter option can be implemented already by the proper output "commands" to RootOutput, but it makes the configuration of the output depend on the presence of specific modules in a specific way (by means of module labels), that is hard to maintain. Better would be if our producer could tell "directly" to RootOutput that it wishes those products lost into oblivion, without RootOutput configuration being pulled in.

Paul Russo and Chris Green proposed a modification to the ProductRegistryHelper::produces() interface allowing an optional additional argument to express the producer's wish about the fate of the product, as "please keep", "please drop" or "whatever". The preference can be overruled by RootOutput if a output command matches the same product, in which case the command will prevail.

The first user of this feature will be LArSoft, to the benefit of LBNE 35t reconstruction code.

History

#1 - 03/04/2015 03:00 PM - Christopher Backhouse

There's a certain pleasing symmetry to giving the second module a line in the constructor like consumes<Hits>(fLabel); to indicate that those Hits shouldn't be in the output event, and perhaps not seen by downstream modules.

This has the nice property that if you do want to seperate the hit producer and consumer into seperate passes (runs of the art executable) you can do so, and the hits will be available in the file for the second job (but then absent again from the final output).

#2 - 03/09/2015 12:17 PM - Christopher Green

- Status changed from New to Accepted

- Estimated time set to 24.00 h

We would like to produce a concrete proposal for this and discuss with experiments how it might (or might not) meet needs in a wider context than your particular case. We will distribute the proposal in email to the stakeholders list.

#3 - 04/17/2015 10:20 AM - Kyle Knoepfel

- Target version set to 521

#4 - 05/04/2015 04:59 PM - Marc Paterno

- Target version changed from 521 to 1.18.00

#5 - 05/15/2015 08:17 AM - Christopher Green

- Due date set to 08/01/2015

#6 - 10/07/2015 03:01 PM - Christopher Green

- Target version changed from 1.18.00 to 834

#7 - 11/03/2015 09:27 AM - Marc Paterno

- Target version changed from 834 to 2.01.00

#8 - 05/06/2016 02:03 PM - Kyle Knoepfel

- Target version changed from 2.01.00 to Vega

#9 - 12/21/2017 01:08 PM - Kyle Knoepfel

- Status changed from Accepted to Assigned

- Assignee set to Kyle Knoepfel

- Target version changed from Vega to 2.10.00

- % Done changed from 0 to 80

#10 - 01/23/2018 12:08 PM - Kyle Knoepfel

- Status changed from Assigned to Resolved

- % Done changed from 80 to 100

#11 - 01/24/2018 07:45 AM - Kyle Knoepfel

- Status changed from Resolved to Closed

This facility has been provided by declaring a produced product via:

```
produces<MyProduct1>("", art::Persistable::No);  // No instance label
produces<MyProduct2>("myInstance", art::Persistable::No);
```

Products that are declared with the art::Persistable::No enumerator **do not** participate in the 'keep' and 'drop' commands supported by output modules. If it is desired to make the persistability configurable, then the author of the module must provide the configuration (e.g.):

```
MyProducer::MyProducer(Parameters const& p)
  : is_persistable_{p().is_persistable() ? art::Persistable::Yes : art::Persistable::No}
{
  produces<MyProduct>("myInstance", is_persistable_);
}
```

where is_persistable_ is of the type art::Persistable. Note that the default art::Persistable value is 'Yes' (e.g.):

```
produces<MyProduct>("myInstance"); // is equivalent to
produces<MyProduct>("myInstance", art::Persistable::Yes);
```

Implemented with commits:

- art:15871ad
- art:32eb607
- canvas:fcbca9e