artdaq Utilities - Feature #21358

DAQInterface should support the concept of subsystems

11/13/2018 09:55 AM - John Freeman

Status:	Resolved	**Start date:**	11/13/2018
Priority:	Normal	**Due date:**	
Assignee:	John Freeman	**% Done:**	90%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Experiment:	-	**Co-Assignees:**	

Description

Eric's been working on the concept of artdaq subsystems, where a subsystem is a collection of artdaq processes which are more tightly coupled to one another than to the processes in another subsystem. He's already developed ways to communicate which processes belong to which subsystem in the boot.txt file on the feature/subsystems branch, but it remains the case that the bookkeeping DAQInterface performs needs to be updated to handle subsystems.

A particular use case he mentioned is that you can have an eventbuilder in subsystem 1 send events not just to dataloggers in subsystem 1, but also to eventbuilders in subsystem 2. More generally, you can have eventbuilders sending events to processes in their subsystem as well as other subsystems.

History

#1 - 11/20/2018 05:03 PM - John Freeman

- Status changed from New to Work in progress

- % Done changed from 0 to 80

Commit comment for 9484a99747a03c59f343bb16a69029560e95a3cf on the feature/subsystems branch:

JCF: update bookkeeping to accommodate simple 2-subsystem arrangements

Since in a commit from earlier today both rank # and subsystem # were added to the artdaq process structure Procinfo, I've rewritten the bookkeeping functionality to take advantage of this so that:

1) artdaq process types within a subsystem bear the same relationship to each other as before (e.g., subsystem 2 dataloggers appear in the destination table for subsystem 2 eventbuilders)

2) An eventbuilder in subsystem 1 adds eventbuilders in subsystem 2 to its destination table, on top of any subsystem 1 dataloggers

3) The routingmaster mediates between subsystem 1 boardreaders and subsystem 1 eventbuilders

I've tested that the sources and destinations tables are constructed correctly (see, e.g., mu2edaq01:/home/jcfree/run_records/1563), and also that the new algorithm is backwards compatible - i.e., if all processes are subsystem 1, then you recover the previous connections between the processes.

The next logical step in the development of subsystem support in DAQInterface, of course, is to generalize this for multiple subsystems.

#2 - 11/24/2018 01:45 PM - John Freeman

In the last couple of days, up through commit b70507c43878afebc48a8a6d893a22272b3dccf7, the following features have been added:

- It's no longer the case that **only** two subsystems, with subsystem 1 sending to subsystem 2, is supported. Instead, the subsystem relationships can be flexibly defined. In particular, to recover the previous behavior, one would add this to the boot file:

```
    Subsystem id: 1
    Subsystem destination: 2

    Subsystem id: 2
```

```
    Subsystem source: 1
```

and then for the processes one would want to put in subsystem 2, one would add process info like the following:

```
EventBuilder host: localhost
EventBuilder label: EventBuilder3
EventBuilder subsystem: 2
```

...where if the subsystem line is left out, subsystem 1 is assumed.

- Each subsystem can have a routingmaster mediating between boardreaders and eventbuilders. If more routingmasters are defined than subsystems in the boot file, an exception is thrown; this is a generalization of DAQInterface's previous behavior, when it would throw an exception if more than one routingmaster was defined.

Probably a good idea at this point would be to add a simple test config which demonstrates subsystem functionality...

#3 - 11/29/2018 11:09 AM - John Freeman

- Status changed from Work in progress to Resolved

- % Done changed from 80 to 90

Since the last comment on this issue, Eric and I have worked on bookkeeping so the following now applies:

- The number of expected fragments per event for an eventbuilder in a given subsystem is derived not just from the boardreaders in the subsystem in question, but also from any subsystems which are part of the subsystem-in-question's ancestral lineage. E.g., if subsystem 3 has two boardreaders sending one fragment per event apiece, and it has as a source subsystem 2 which has three boardreaders sending two fragments per event apiece, which in turn has as a source subsystem 1 with one boardreader sending three fragments apiece, then DAQInterface will set expected_fragments_per_event to 11 during bookkeeping

- A similar logic applies when DAQInterface performs memory management. Say we're not using advanced memory usage, the max fragment size is set to 2 megabytes in the DAQInterface settings file, and max_event_size_bytes isn't set in a subsystem 3 eventbuilder's FHiCL document (which would be an override); in that case, DAQInterface sets the buffer size between subsystem 3 eventbuilders and dataloggers to 22 megabytes .

- In the case that advanced memory usage IS used, the only wrinkle on top of advanced memory usage's behavior in the traditional single-subsystem case is that memory for events is only scaled up by 10% **once** regardless of the number of cascading subsystems. For the sake of simplicity, let's say that max_fragment_size_bytes is set to 2 megabytes in all the boardreaders across all three subsystems. Then while the eventbuilders in subsystem 1 will have buffer sizes of 3 * 2 * 1.1 = 6.6 megabytes for their downstream transfer plugins, the outgoing transfer plugin size buffer for the eventbuilder in subsystem 2 won't be (6.6 megabytes + 3*2*2 megabytes)*1.1 = 20.46 megabytes, but rather, 6.6 megabytes + 3*2*2 megabytes * 1.1 = 19.8 megabytes

The changes have been tested using the simple_subsystems config; see e.g. mu2edaq01:/home/jcfree/run_records/1601 (non-advanced memory usage) and 1602 (advanced).

#4 - 11/29/2018 03:07 PM - John Freeman

- Status changed from Resolved to Work in progress

For the record, the above comment I made earlier today applies to develop branch commit fdcfff9b1510c3bce73bf41e60a88ad04d624bb8, with the immediately-preceding commit 12a0c7cd329d9a265b5f6fc2be696c195bc5bbdf already containing all the up-to-date bookkeeping code.

#5 - 02/18/2019 02:43 PM - John Freeman

- Status changed from Work in progress to Resolved

A few days ago Eric and I agreed that subsystem implementation had been matured to the point that reviews could begin; hence I've marked this "Resolved".