artdaq Utilities - Bug #21611

DAQInterface advanced memory usage seems to have trouble with certain FHiCL over-rides

01/02/2019 08:51 AM - Kurt Biery

Status:	New	**Start date:**	01/02/2019
Priority:	Normal	**Due date:**	
Assignee:	John Freeman	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Experiment:	-	**Co-Assignees:**	

Description

To reproduce the problem:

- either use an existing artdaq-demo installation or create a new one, and update the artdaq-utilities-daqinterface package to commit hash ac86bcb0037e6968df3fd52b59ac80186ad86a59
- run 'sh ./run_demo.sh --config mediumsystem_with_routing_master --bootfile `pwd`/artdaq-utilities-daqinterface/simple_test_config/mediumsystem_with_routing_master/boot.txt --comps component01 component02 component03 component04 component05 component06 component07 component08 component09 component10 --runduration 40 --no_om'
- change the artdaq-utilities-daqinterface commit hash to a0a50a104d119be8c336a9d81af4fe381bdb9fd4 and re-run the test listed above

(Both of these commits were done on the 'develop' branch of artdaq-utilities-daqinterface.)

Then,

- compare the differences in those two versions of the code. As you will see, the only difference is the presence of the component01_standard.fragment_receiver.max_fragment_size_bytes parameter in component_standard.fcl in the earlier commit. Please Note that this parameter is not used at this point in time. component01_hw_cfg.fcl does not yet include component_standard.fcl, etc.
- compare the two sets of run_records for the two runs that were just performed. As you will see, the second run has incorrect values for the max_fragment_size_words parameter in the component02-10 FHiCL files. (It has 1024, when it should have 128000.)

It seems as though the advanced memory usage logic is picking up the last value for max_fragment_size_bytes in component_standard.fcl and using it for all components, even though that declaration of max_fragment_size_bytes is specific to component01.

For reference, here is a copy of the settings_example file that I am using:

```
[biery@mu2edaq01 DAQInterface]$ pwd
/home/biery/331Demo/DAQInterface

[biery@mu2edaq01 DAQInterface]$ cat settings_example

# JCF, Sep-16-2017

# This file is an example of a settings file which would work with an
# artdaq-demo installation, assuming the installation was performed
# with artdaq-demo's quick-mrb-start.sh script. It is valid as of
# artdaq-demo v2_10_02; more details on artdaq-demo installation can
# be found in
# https://cdcvs.fnal.gov/redmine/projects/artdaq-demo/wiki. Note that
# the user will need to (A) make sure that a directory called
# $HOME/run_records has been created and (B) make sure to set the
# productsdir_for_bash_scripts variable, below, to a products
# directory containing the xmlrpc_c package (needed for DAQInterface
# to receive commands)

log_directory: /home/biery/331Demo/daqlogs
data_directory_override: /scratch/biery/data
```

```
record_directory: /home/biery/331Demo/run_records
package_hashes_to_save: [ artdaq-demo, artdaq-core-demo, artdaq ]
productsdir_for_bash_scripts: /cvmfs/fermilab.opensciencegrid.org/products/artdaq

boardreader timeout: 60
eventbuilder timeout: 30
aggregator timeout: 30

# Currently (as of 2018-07-10) needs to be big enough for all simple_test_config/ examples.
# The "biggest" example has 10 BR's -- so the xfer between the EB and DL needs to be able
# to handle the data from the 10 BRs.
#max_fragment_size_bytes: 91000000

all_events_to_all_dispatchers: true

advanced_memory_usage: true

transfer_plugin_to_use: TCPSocket

[biery@mu2edaq01 DAQInterface]$ date
Wed Jan  2 08:49:47 CST 2019
```

History

#1 - 01/08/2019 06:16 PM - John Freeman

I'll start with the reason for the problem; then I'll discuss a couple of solutions:

The issue is that it's DAQInterface's code, and not a FHiCL interpreter, which is plucking out the max_fragment_size_bytes variable from the FHiCL document, so some of the capabilities of a FHiCL interpreter - in the example given above, the ability to associate variables with certain tables - are missing. DAQInterface simply looks for the last (uncommented) line in the FHiCL document which contains a max_fragment_size_bytes variable and uses its value, i.e.

```
res = re.findall(r"\n[^#]*max_fragment_size_bytes\s*:\s*([0-9\.exabcdefABCDEF]+)", procinfo.fhicl_used)
```

...meaning that it's not sophisticated enough to figure out that a line like

```
component01_standard.fragment_receiver.max_fragment_size_bytes: 8192
```

at the end of a FHiCL document is only relevant to component01, and not component02, etc.

Possible solutions include:

- Have DAQInterface call fhicl-dump on the FHiCL documents BEFORE performing bookkeeping, rather than after, as is currently the case. If this change is made, fhicl-dump can process out the syntactical complexity of a line like "component01_standard.fragment_receiver.max_fragment_size_bytes: 8192", providing DAQInterface with the **correct** max_fragment_size_bytes value in a simple "max_fragment_size_bytes: <value>" format. However, once this happens, in order to preserve the "canonicalness" of the FHiCL documents we're either stuck with (A) rerunning fhicl-dump again after bookkeeping is performed, or (B) attempting to get bookkeeping to preserve the canonicalness of the FHiCL document, which might be difficult especially in parts of bookkeeping where variables get added into the FHiCL (e.g., max_event_size_bytes getting added into the eventbuilder, datalogger and dispatcher processes automatically in the case of advanced memory management - DAQInterface would need to figure out alphabetically where to place the variable, preserve whitespace correctly, etc.)
- Have DAQInterface not allow you to use any FHiCL syntax more sophisticated than a "max_fragment_size_bytes: <value>" on a line, i.e., insist there only be whitespace between the start of the line and the "max_fragment_size_bytes" token. If we do that, we lose some of the sophistication of FHiCL.
- Have DAQInterface call some Python function provided by FHiCL developers which will give it the correct max_fragment_size_bytes value once the function is provided with the FHiCL document. The disadvantage there is that we'd be adding an additional dependency to DAQInterface if this were provided in some package.