artdaq Utilities - Idea #10600

Store the configurations of each artdaq process in the art/ROOT data file(s)

10/20/2015 05:24 PM - Kurt Biery

Status:	Closed	**Start date:**	10/20/2015
Priority:	Normal	**Due date:**	
Assignee:	John Freeman	**% Done:**	100%
Category:	artdaq-daqinterface	**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Experiment:	-		

Description

History

#1 - 08/12/2016 11:33 AM - Kurt Biery

- Subject changed from Store the configurations of each artdaq process in a run-based data product for each run to Store the configurations of each artdaq process in the art/ROOT data file(s)

Here is a copy of an email exchange on this topic from April 18,19 2016:

Kurt wrote:

Wow, cool.
I needed to tell it to print out the "process" configuration (-P) instead of the module configuration (-M [default]), but I think that I got it to print "everything":

```
[biery@mu2edaq01 tmp]$ config_dumper -P artdaqdemo_r000418_sr01_20160418T210158_mod3.root
=============================================
Processing file: artdaqdemo_r000418_sr01_20160418T210158_mod3.root

    1. Read SQLiteDB from file, total size: 16.0 KiB.

DAQAG: {
BoardReader_TOY1_mu2edaq01_6305: {
daq: {
fragment_receiver: {
board_id: 0
distribution_type: 1
event_builder_count: 2
first_event_builder_rank: 2
fragment_id: 0
fragment_type: "TOY1"
generator: "ToySimulator"
mpi_buffer_count: 16
mpi_sync_interval: 50
nADCcounts: 1e6
random_seed: 9868
sleep_on_stop_us: 500000
throttle_usecs: 100000
}
max_fragment_size_words: 2.097152e6
metrics: {
brFile: {
fileName: "/tmp/boardreader/br_%UID%_metrics.log"
level: 3
metricPluginType: "file"
uniquify: true
}
}
}
}
daq: {
```

aggregator: {
event_builder_count: 2
event_queue_depth: 20
event_queue_wait_time: 5
expected_events_per_bunch: 1
file_duration: 0
file_event_count: 0
file_size_MB: 8000
first_event_builder_rank: 2
mpi_buffer_count: 8
onmon_event_prescale: 1
print_event_store_stats: true
xmlrpc_client_list: ";
http://mu2edaq01.fnal.gov:6305/RPC2.3;http://mu2edaq01.fnal.gov:6306/RPC2.3;http://mu2edaq01.fnal.gov:6335/RPC2.4;http://mu2edaq01.fnal.gov:6336/RPC2.4;http://mu2edaq01.fnal.gov:6365/RPC2.5;http://mu2edaq01.fnal.gov:6366/RPC2.5"
}
max_fragment_size_words: 2.097152e6
metrics: {
aggFile: {
fileName: "/tmp/aggregator/agg_%UID%_metrics.log"
level: 3
metricPluginType: "file"
uniquify: true
}
}
}
outputs: {
normalOutputMod2: {
SelectEvents: {
SelectEvents: [
"pmod2"
]
}
compressionLevel: 0
fileName: "/home/biery/tmp/artdaqdemo_r%06r_sr%02s_%to_mod2.root"
module_label: "normalOutputMod2"
module_type: "RootOutput"
}
normalOutputMod3: {
SelectEvents: {
SelectEvents: [
"pmod3"
]
}
compressionLevel: 0
fileName: "/home/biery/tmp/artdaqdemo_r%06r_sr%02s_%to_mod3.root"
module_label: "normalOutputMod3"
module_type: "RootOutput"
}
}
physics: {
analyzers: {}
filters: {
prescaleMod2: {
module_label: "prescaleMod2"
module_type: "NthEvent"
nth: 2
}
prescaleMod3: {
module_label: "prescaleMod3"
module_type: "NthEvent"
nth: 3
}
}
my_output_modules: [
"normalOutputMod2",
"normalOutputMod3"
]
p1: []
p2: [
"BuildInfo"
]
pmod2: [
"prescaleMod2"
]

pmod3: [
"prescaleMod3"
]
producers: {
BuildInfo: {
instance_name: "ArtdaqDemo"
module_label: "BuildInfo"
module_type: "ArtdaqDemoBuildInfo"
}
}
}
services: {
CatalogInterface: {
service_provider: "TrivialFileDelivery"
service_type: "CatalogInterface"
}
FileTransfer: {
service_provider: "TrivialFileTransfer"
service_type: "FileTransfer"
}
NetMonTransportServiceInterface: {
max_fragment_size_words: 2.097152e6
service_provider: "NetMonTransportService"
service_type: "NetMonTransportServiceInterface"
}
message: {
destinations: {
STDOUT: {
categories: {
ArtReport: {
limit: 100
}
default: {
limit: -1
}
}
threshold: "INFO"
type: "cout"
}
}
}
scheduler: {
errorOnFailureToPut: false
fileMode: "NOMERGE"
}
}
source: {
module_label: "source"
module_type: "NetMonInput"
}
}

On 4/18/16 4:17 PM, Wesley Ketchum wrote:

> Hi Kurt,
>
> Out of curiosity, can you run an "config_dumper" command on the output art file? That should print a number of things out too (but perhaps not the BoardReader pieces ... could be a request to artists)
>
> Wes
>
> On Mon, Apr 18, 2016 at 3:14 PM Kurt A Biery <biery@fnal.gov> wrote:
>
> Hi everyone,
> When we last talked about storing CFG information in ROOT files, it was suggested that simply passing a system-wide FHiCL document to the disk-writing Aggregator would be enough to get art to do this automatically. Mainly out of curiosity, I've verified this to be true.
>
> The code snippet that I used to pull out the DAQAG parameter set in an offline EDAnalyzer "analyze" method is the following:
>
> ```
> fhicl::ParameterSet agPSet;
> bool retc = evt.getProcessParameterSet("DAQAG", agPSet);
> if (retc) {std::cout << "PSET " << agPSet.to_string() << std::endl;}
> else {std::cout << "Process parameter set not found." << std::endl;}
> ```
>
> When I run this on a data file generated in an artdaq-demo test run in which I buggered the AG FHiCL file to have an extra "BoardReader"

section, I get the following from the above code (the "extra" BR stuff is high-lighted in orange):

PSET BoardReader_TOY1_mu2edaq01_6305:{daq:{fragment_receiver:{board_id:0 distribution_type:1 event_builder_count:2 first_event_builder_rank:2 fragment_id:0 fragment_type:"TOY1" generator:"ToySimulator" mpi_buffer_count:16 mpi_sync_interval:50 nADCcounts:1e6 random_seed:9868 sleep_on_stop_us:500000 throttle_usecs:100000} max_fragment_size_words:2.097152e6 metrics:{brFile:{fileName:"/tmp/boardreader/br_%UID%_metrics.log" level:3 metricPluginType:"file" uniquify:true}}}} daq:{aggregator:{event_builder_count:2 event_queue_depth:20 event_queue_wait_time:5 expected_events_per_bunch:1 file_duration:0 file_event_count:0 file_size_MB:8000 first_event_builder_rank:2 mpi_buffer_count:8 onmon_event_prescale:1 print_event_store_stats:true xmlrpc_client_list:"; http://mu2edaq01.fnal.gov:6305/RPC2,3;http://mu2edaq01.fnal.gov:6306/RPC2,3;http://mu2edaq01.fnal.gov:6335/RPC2,4;http://mu2edaq01.fnal.gov:6336/RPC2,4;http://mu2edaq01.fnal.gov:6365/RPC2,5;http://mu2edaq01.fnal.gov:6366/RPC2,5"} max_fragment_size_words:2.097152e6 metrics:{aggFile:{fileName:"/tmp/aggregator/agg_%UID%_metrics.log" level:3 metricPluginType:"file" uniquify:true}}} outputs:{normalOutputMod2:{SelectEvents:{SelectEvents:["pmod2"]} compressionLevel:0 fileName:"/home/biery/tmp/artdaqdemo_r%06r_sr%02s_%to_mod2.root" module_label:"normalOutputMod2" module_type:"RootOutput"} normalOutputMod3:{SelectEvents:{SelectEvents:["pmod3"]} compressionLevel:0 fileName:"/home/biery/tmp/artdaqdemo_r%06r_sr%02s_%to_mod3.root" module_label:"normalOutputMod3" module_type:"RootOutput"}} physics:{analyzers:{} filters:{prescaleMod2:{module_label:"prescaleMod2" module_type:"NthEvent" nth:2} prescaleMod3:{module_label:"prescaleMod3" module_type:"NthEvent" nth:3}} my_output_modules:["normalOutputMod2","normalOutputMod3"] p1:[] p2:["BuildInfo"] pmod2:["prescaleMod2"] pmod3:["prescaleMod3"] producers:{BuildInfo:{instance_name:"ArtdaqDemo" module_label:"BuildInfo" module_type:"ArtdaqDemoBuildInfo"}}} process_name:"DAQAG" services:{CatalogInterface:{service_provider:"TrivialFileDelivery" service_type:"CatalogInterface"} FileTransfer:{service_provider:"TrivialFileTransfer" service_type:"FileTransfer"} NetMonTransportServiceInterface:{max_fragment_size_words:2.097152e6 service_provider:"NetMonTransportService" service_type:"NetMonTransportServiceInterface"} message:{destinations:{STDOUT:{categories:{ArtReport:{limit:100} default:{limit:-1}} threshold:"INFO" type:"cout"}}} scheduler:{errorOnFailureToPut:false fileMode:"NOMERGE"}} source:{module_label:"source" module_type:"NetMonInput"}

As we discussed earlier, this gives support to a model in which the FHiCL document that we send to each artdaq process has the full configuration information and each process knows how to pull out its sub-section. Of course, we should verify that this doesn't become too unwieldy in actual practice (e.g. lots of detector parameters for each BoardReader).

Just an FYI,
Kurt

#2 - 08/12/2016 11:35 AM - Kurt Biery

- File artdaqMeeting_Whiteboard_10Aug2016_SingleFCL_Ideas.jpg added

Attaching a picture of the whiteboard from our meeting on 10-Aug-2016 in which we talked about how to structure the FHiCL document that we send to the *artdaq* process(es) that write data file(s).

#3 - 08/12/2016 01:21 PM - Kurt Biery

- Assignee set to John Freeman

#4 - 08/12/2016 05:54 PM - John Freeman

- File run_19391_config_dumper_example.txt added

- % Done changed from 0 to 80

Modifications performed on DAQInterface (available for commit cadb98b22058fed33c469b1ab329c703c8198757 of lbnerc) now mean that when a root file is written out by a diskwriting aggregator, if one runs "config_dumper -P <rootfile>" there's a table within called "run_documents" which contains the FHiCL documents used by every artdaq process in the run, as well as the DAQInterface configuration file and the run output metadata file, stored in strings labeled with "run_daqinterface_config" and "run_metadata", respectively. Additionally, if RCEs were in the run, the XML file used to configure them would be saved in a string labeled by "run_rce_xml". An example of this can be found in the attached document, "run_19391_config_dumper_example.txt". Obviously, different experiments will have different files saved, but it's good to know that an entire document can be stored within a string.

#5 - 08/15/2016 01:46 PM - John Freeman

On lbne35t-gateway01, I've added a script to /data/lbnedaq/bin called "config_viewer.sh". Running it is simple - just pass it the root file output by the DAQ, and it will print the configuration contents, both the FHiCL as well as the additional configuration data described in the last entry. This script serves as a wrapper around config_dumper, and additionally will properly handle the newlines embedded in the strings used to store the configuration data. To see an example of the output, try running

```
config_viewer.sh /data/lbnedaq/scratch/jcfree/Documents/rootfiles/lbne_r019407_sr01_20160815T174741_0.root
```

#6 - 01/13/2017 03:57 PM - Eric Flumerfelt

- Category set to Additional Functionality

- Target version deleted (577)

#7 - 01/27/2017 02:28 PM - John Freeman

- Status changed from New to Resolved

- % Done changed from 80 to 100

The new, standard DAQInterface (https://cdcvs.fnal.gov/redmine/projects/artdaq-utilities/wiki/Artdaq-daqinterface) contains the same functionality as its 35ton predecessor described in previous entries for this Redmine Issue. Additionally, as a crosscheck, the "./bin/compare_run_record_and_rootfile.sh" script has been added - taking a run #, this script will compare the DAQInterface configuration file and the metadata file saved in run records with the DAQInterface configuration file and metadata file saved as FHiCL string values in the *.root files for that run, telling you whether or not there's the expected agreement. This crosscheck is performed in the ./bin/just_do_it.sh script automatically, and can be performed manually by users as well. Consistent agreement has been observed.

#8 - 02/10/2017 01:03 PM - Eric Flumerfelt

- Project changed from artdaq to artdaq Utilities

- Category changed from Additional Functionality to 431

#9 - 02/10/2017 01:07 PM - Eric Flumerfelt

- Category changed from artdaq-utilities to artdaq-daqinterface

#10 - 05/22/2017 08:37 AM - Eric Flumerfelt

- Status changed from Resolved to Closed

Closing, as this functionality has been available for some time.

Files

artdaqMeeting_Whiteboard_10Aug2016_SingleFCL_Ideas.jpg	3.12 MB	08/12/2016	Kurt Biery
run_19391_config_dumper_example.txt	24.3 KB	08/12/2016	John Freeman