artdaq - Feature #20580

Bookkeep max_event_size_bytes to reflect different fragment sizes from different fragment generators

08/09/2018 10:22 PM - John Freeman

Status:	Assigned	**Start date:**	08/09/2018
Priority:	Normal	**Due date:**	
Assignee:	John Freeman	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
Experiment:	-	**Co-Assignees:**	

Description

Right now, in SharedMemoryEventManager, the max_event_size_bytes parameter must either be set explicitly in the the FHiCL code or default to expected_fragments_per_event*max_fragment_size_bytes. The issue with setting max_event_size_bytes explicitly in a configuration is that different runs typically have different numbers and types of fragment generators, resulting in a *true* max event size which varies from run to run. The issue with going with the default of expected_fragments_per_event*max_fragment_size_bytes is that even if you know how many fragments per event there are, typically the max size of each fragment will differ for different types of fragment generator.

It would be useful, then, if there were an option for DAQInterface to calculate on-the-fly what the max_event_size_bytes should be, and insert that value in the configuration during bookkeeping. It could require that the FHiCL documents for boardreaders explicitly state up front what the max fragment size for their particular fragment generators were, and then it could sum the max fragment sizes across all the boardreaders in the run to set max_event_size_bytes accordingly, with some "headroom" added.

History

#1 - 08/15/2018 12:55 PM - Kurt Biery

This is probably already planned, but we should move to a model in which "max_fragment_size_bytes" is specified in BoardReader configuration files and "max_event_size_bytes" is specified in EventBuilder, DataLogger, and Dispatcher config files. (I think that the only change here is moving away from using max_fragment_size_bytes in EB, DL, and Dispatcher config files.)

#2 - 08/17/2018 02:17 PM - Kurt Biery

- Status changed from New to Assigned

#3 - 08/20/2018 01:22 PM - John Freeman

- Status changed from Assigned to Resolved

- % Done changed from 0 to 100

This is now available on the develop branch, commit 7a09986462371be841b58e2a3ef3fed2c604feb8. If the line advanced_memory_usage: true is added to the file referred to by DAQINTERFACE_SETTINGS, then DAQInterface will:

1) Require that all boardreader FHiCL documents, and no other type of FHiCL document, contain a parameter called "max_fragment_size_bytes"

2) Sum over the values of max_fragment_size_bytes to set max_event_size_bytes in the non-boardreader FHiCL documents. It will either clobber existing values of max_event_size_bytes, or, if the parameter isn't present, insert it into the FHiCL document.

#4 - 08/22/2018 12:07 PM - Kurt Biery

- Status changed from Resolved to Assigned

I ran some tests of this new code in the artdaqDemo.

There were a couple of minor tweaks that I committed to the feature/tweaks_to_advanced_mem_usage branch that I needed to get things to work with the mediumsystem_with_routing_master sample configuration. John, please take a look and verify that they can be merged to the develop branch.

Also, I noticed that the 10% safety factor is being applied to both the BoardReader max_fragment_size_bytes and the derived EventBuilder, etc. max_event_size_bytes. Is that by-design?
Thanks,
Kurt

#5 - 08/23/2018 06:03 AM - John Freeman

I think the removal of the requirement that the RoutingMaster process contain a max event size is eminently reasonable, and I'm glad you fixed my blunder where max_fragment_size_bytes had been required to be set in the settings file when advanced memory usage was true (rather than false, as was supposed to be the case). Apologies for my failure to catch this in testing.

Concerning the 10% safety factor: right now it's applied to each fragment's input max_fragment_size_bytes, and the resulting max event size is the sum of those - i.e., 1.1*(first fragment's max bytes + second fragment's max bytes + ...). That seems like what we'd want, though perhaps I'm underthinking it- let me know if you have another proposed approach.

Concerning the assertion failure that would occur if DAQInterface couldn't find "buffer_count" in the FHiCL document: rather than saying, e.g., "(the 'buffer_count' parameter needs to be added to the EventBuilderMain configuration document)", I'd suggest something like "DAQInterface code needs to be changed to reflect that buffer_count is apparently no longer required in EventBuilderMain configuration document"

#6 - 08/25/2018 10:19 AM - John Freeman

Made some changes with commit db91da6a8bcf43ee6ec5da65ed35d62649ab3e1e; I'll reprint the commit comments here:

Couple of improvements to DAQInterface when run with advanced memory management:

- Improve the regexp match so that it ignores max_fragment_size_bytes and max_event_size_bytes in FHiCL documents when they're in commented-out lines
- If DAQInterface can't find the line with buffer_count under which to insert max_event_size_bytes, emphasize that this is due to faulty assumptions made by DAQInterface, not because the FHiCL document's supposed to have this parameter. This way, DAQInterface depends on artdaq, but not vice-versa.
- If advanced memory management is set to true in the settings file, make it an error if max_fragment_size_bytes is also set - while DAQInterface can ignore this parameter, it's very misleading to have it in there.

#7 - 08/26/2018 09:38 AM - John Freeman

Performed further testing of the code, and found that it wasn't accurately determining max_fragment_size_bytes in the BoardReader FHiCL document if either

(A) max_fragment_size_bytes appeared more than once (it had been picking the value from the first appearance, rather than the last, which is what we want)
(B) max_fragment_size_bytes didn't have only whitespace between it and the start of the line, so it was missing out on cases like "daq.fragment_receiver.max_fragment_size_bytes: 10000000"

Both issues above have been fixed with develop branch commit 231d7633aeafe0e1d69f7e24b48fe5591dcc1f23, and the fixes also appear on the feature/protodune branch.

#8 - 08/28/2018 09:16 AM - Kurt Biery

Regarding whether to apply the 10% safety factor to BR max_fragment_size_bytes parameters... At a minimum, this should be discussed with the group. One of the primary places where this will be used is in the Transfer Plugin buffer size. We've had a lot of discussions recently about tuning the size of the TCP send buffer size (which is set to the Transfer Plugin buffer size), and my sense is that it will be very confusing to set a BR max_fragment_size_bytes to a certain value and have the TCP send buffer size set to a value that is 10% larger.