

## artdaq - Idea #14781

### Investigate starting art as a subordinate, but separate, process in the EventBuilder and Aggregator

12/07/2016 03:44 PM - Kurt Biery

<b>Status:</b>	Closed	<b>Start date:</b>	12/07/2016
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assignee:</b>	Eric Flumerfelt	<b>% Done:</b>	0%
<b>Category:</b>	Additional Functionality	<b>Estimated time:</b>	0.00 hour
<b>Target version:</b>	artdaq v3_00_01		
<b>Experiment:</b>	-		
<b>Description</b>			
<p>I just had a discussion with Kyle Knoepfel, Chris Green, and Paul Russo about reconfiguring <i>art</i> in a DAQ context. (This is related to Issue <a href="#">#3190</a>.) Nothing has changed within <i>artdaq</i> or <i>art</i> since we last discussed this issue, and the following are still true:</p> <ul style="list-style-type: none"><li>• <i>artdaq</i> processes start an <i>art</i> thread when they receive their first "init" command and leave it running for the rest of the DAQ session</li><li>• there are ways to selectively enable and disable <i>art</i> modules, and there is the possibility to feed selected configuration changes to an output module, but free-form reconfiguration of an <i>art</i> thread is not currently available</li><li>• stopping the <i>art</i> thread at end-run time and starting a new thread at begin-run time is not currently an option since there are various types of information that are stored in singletons in <i>art</i> that would not be cleared out of the <i>process</i> memory if the thread is restarted<ul style="list-style-type: none"><li>◦ it was noted that it is possible to imagine changes to the components of <i>art</i> that store this information so that they store it differently and/or completely clean it up if/when an <i>art</i> thread is restarted. However, this would need to be a specific request to the <i>art</i> team.</li></ul></li><li>• a separate <i>art</i> process that is started at begin-run time and shut down and end-run time seems to be the most flexible way of supporting free-form reconfiguration of <i>art</i> within <i>artdaq</i></li></ul> <p>Given the recent additions to our data-transfer possibilities (e.g. SHM), it now seems reasonable to consider replacing the <i>art</i> threads in the EB and AG with child processes. We should look into this.</p> <p>Along with the ability to support free-form reconfiguration of <i>art</i> between runs, there could be additional benefits of having separate <i>art</i> processes like having the possibility of separate execution environments for an EB and its associated <i>art</i> process.</p>			
<b>Related issues:</b>			
Related to artdaq - Idea #15350: art as separate process in EventBuilder and ...		<b>Closed</b>	<b>01/27/2017</b>

## History

### #1 - 01/13/2017 11:10 AM - Kurt Biery

As everyone on the watch list for this Issue probably recalls, EricF expressed some concern about the performance impact that extra copies to/from shared memory would cause if we are to switch to a model that has an *art* sub-process rather than an *art* thread. Based on that, I went back to talk with MarcP, and here are some notes from that discussion.

It's true that some of the limitations in *art* related to being unable to stop and restart an *art* thread in a single instance of a process will be fixed as part of the work on a multi-threaded version of *art*. However, Marc shared concerns that ROOT has possibly worse issues with static initialization that could only be safely avoided if we excluded **all** use of ROOT in our *art* threads. This would need to include ROOT output as well as any use of ROOT utilities in user modules. Marc imagines that someday there may be a version of *art* that is free of all dependencies on ROOT, and this would allow us to enforce that condition, but that is far in the future.

So, Marc is still advocating an *art* sub-process. Of course, he and JimK are open to talking about ways to make the data transfer efficient, and we should set up some discussions if that would be helpful. One cool technology that Jim mentioned for the future is Intel 3D XPoint. It sounds like this will allow memory to be shared among processes (among other features).

For reference, I want to make a few notes on what is meant by multi-threaded *art*. The first step in this work is to support event-based parallelism. That means that a single Linux process can run multiple *art* threads, each of which processes a separate event. Within each thread, the event processing is essential linear, like what we're used to now. In terms of overall throughput, this is fairly equivalent to running multiple *art* processes, each of which is fed a different set of events. However, the advantages of having an event-based MT *art* include

- reducing the overall amount of memory used on the computer to store job-wide metadata such as calibration constants and detector geometry
- reducing the stress on grid-based batch submission schemes, which may work fine with processing runs of N jobs, but might run into problems with 10\*N jobs (where N can be 10,000..100,000)
- reducing the amount of work needed to merge the output files from many parallel processes

The second step in the MT *art* work is to support the parallel running of modules within a single *art* thread. So for example, the calorimeter data-unpacking module could run in parallel with the tracker unpacking module (since there work is independent). The expectation is that the

event-based parallelism will provide the most benefit (~80-90%), so that is why it is being done first.

**#2 - 01/13/2017 03:50 PM - Eric Flumerfelt**

- *Category set to Additional Functionality*
- *Target version deleted (577)*

**#3 - 02/03/2017 09:23 AM - Eric Flumerfelt**

- *Related to Idea #15350: art as separate process in EventBuilder and DataLogger added*

**#4 - 01/27/2018 12:33 PM - Eric Flumerfelt**

- *Status changed from New to Closed*
- *Assignee set to Eric Flumerfelt*
- *Target version set to artdaq v3\_00\_01*