artdaq Utilities - Feature #22061

DAQInterface should algorithmically determine whether an artdaq process is critical

03/05/2019 12:37 PM - John Freeman

Status:	Reviewed	**Start date:**	03/05/2019
Priority:	Normal	**Due date:**	
Assignee:	John Freeman	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Experiment:	-	**Co-Assignees:**	Eric Flumerfelt

Description

Thanks to Issue #21695, at the current head of develop DAQInterface, when run in direct process management mode, will not stop running if a process dies or throws an exception unless that process's label is on a critical list. However, after discussion at yesterday's artdaq meeting, we agreed that it would make more sense if DAQInterface decided on a process-by-process basis whether or not the process should default to critical. For example, by default all boardreaders would be critical, while dispatchers wouldn't be. Eventbuilders would or wouldn't be critical as a function of how many were used in a run. If an artdaq process died or entered an error state, DAQInterface would not only print this info to screen (as it already does), but also announce why it was making the decision about criticality it made, and how users could override it.

History

#1 - 03/14/2019 01:35 PM - John Freeman

A further discussion on this topic occurred at this past Monday's artdaq meeting, and it concerned overriding DAQInterface's algorithmic decisions about whether or not a process was critical. Essentially, the question is: what interface do we provide the DAQInterface user with in order for them to override DAQInterface's default decisions? Here's one idea: we could have a file (referred to by, say, $DAQINTERFACE_CRITICAL_PROCESS_OVERRIDES) formatted like the following:

```
<regular expression covering process labels> <minimum fraction of original processes with these labels we need for a run>
```

so, e.g., if the $DAQINTERFACE_CRITICAL_PROCESS_OVERRIDES file had these two lines:

```
EventBuilder.* 0.5
DataLogger.* 1.0
```

we'd be telling DAQInterface that it should end the run after we've lost at least half our EventBuilders, or if ANY DataLoggers die.

Please let me know if you think this is a good idea, or if you have something else in mind, and let's see if we can converge.

#2 - 03/19/2019 09:53 AM - John Freeman

More discussion took place at the meeting yesterday. It was agreed that on each line of the critical process override file, along with the process label regex and the fraction of required processes, there should be a third field, the minimum absolute number of processes whose labels match the regex. So, e.g.:

```
EventBuilder.* 0.5 2
```

would mean "For the run to not be ended, I require that at least half the processes whose label begins with EventBuilder still be running, AND that there be at least 2 such processes".

In general, we'd "AND" the requirements on each line. E.g.:

```
EventBuilder.* 0.5 2
EventBuilder2 1.0 0
```

...would mean "I don't want the run to continue if I don't have at least half the original eventbuilders (of which there were at least two), or if, in a run which includes EventBuilder2, we lose that particular eventbuilder".

However the question of overrides comes up. E.g., if someone writes:

```
EventBuilder2 1.0 0
```

```
EventBuilder2 0 1
```

should this (A) be an error, or (B) result in the first line being ignored, or (C) just result in the two lines being AND-ed, so that we'd both require there be a process with label EventBuilder2 (line 2) and require that it not die (line 1)?

#3 - 03/19/2019 10:46 PM - John Freeman

- % Done changed from 0 to 100

- Status changed from New to Resolved

With commit 5b998ffd6e17b5aa78f0715fee67e55937887e0a on feature/issue22061_critical_processes, the following applies:

If the environment variable DAQINTERFACE_PROCESS_REQUIREMENTS_LIST is set when DAQInterface is launched, DAQInterface interpret that environment variable as referring to a file which contains a set of rules for when process death (or Error state) should end a run. DAQInterface expects each line in $DAQINTERFACE_PROCESS_REQUIREMENTS_LIST to contain three fields:

- A regex to which it can try to match the deceased process's label
- A fraction representing the fraction of processes which matched the regex at run start which have to still exist for the run not to end
- A number representing the raw number of processes which match the regex which have to still exist for the run not to end.

So, e.g., the line

```
component.* 0.5 1
```

means that there both has to be at least half as many processes whose labels begin with "component" as the run started out with, and at least one such process in an absolute sense, during the run - otherwise DAQInterface will end the run. So if we performed a run with two boardreaders, component01 and component02, you could afford to lose one of them. On the other hand, if we changed the line to look like this:

```
component.* 0.51 1
```

or this:

```
component.* 0.5 2
```

then if one of the two boardreaders died you'd see something like the text below, followed by DAQInterface returning everything to the "Stopped" state:

```
Error: loss of process component02 drops the total number of processes
whose labels match the regular expression "component.*" to 1 out of an
original total of 2; this violates the minimum number of 2 required in the
file "/tmp/process_requirements_list.txt"
```

(where here it's component02 which died, and DAQINTERFACE_PROCESS_REQUIREMENTS_LIST is set to /tmp/process_requirements_list.txt).

Note that it is **not** required that all processes in a run match any of the regexes in $DAQINTERFACE_PROCESS_REQUIREMENTS_LIST. If this is not the case - e.g., we have a one line file which covers component boardreaders but says nothing about other processes - then DAQInterface determines whether a process death or Error state should end the run by the following simple criteria:

- If a process is a boardreader, end the run
- If the process writes data to disk (whether eventbuilder or datalogger), end the run
- All other processes, just print a warning and continue with the run

Note that if a process label matches one or more regexes in $DAQINTERFACE_PROCESS_REQUIREMENTS_LIST, these default rules are ignored.

#4 - 03/20/2019 03:40 PM - Eric Flumerfelt

There should be an example DAQINTERFACE_PROCESS_REQUIREMENTS_LIST in the docs directory. This could serve a documentation for the fallback set of rules...
default_process_requirements never appears to be set or checked.

#5 - 03/21/2019 05:09 PM - John Freeman

The head of feature/issue22061_critical_processes, b64e8a7709a469e44ace8c361bfe128ea62aabea, now implements Eric's suggestions. The name of the example file is docs/process_requirements_list_example. In addition to this, Eric mentioned to me offline that the default requirement for eventbuilders should be that there's at least one, so that's been implemented as well at the head.

#6 - 03/24/2019 02:15 PM - John Freeman

A note to the reviewer: this branch can't currently be merged into develop without conflicts, since there's logic from

feature/issue21919_dont_start_run_if_problem in manage_processes_direct.py which is based on the v3_04_00 model of a critical process list. After testing this branch, notify me and I'll merge it into develop.

#7 - 03/25/2019 09:36 AM - Eric Flumerfelt

- Status changed from Resolved to Reviewed

- Co-Assignees Eric Flumerfelt added

The feature works as advertised. One final comment: the message that prints when DAQInterface is performing a default action:

```
Error: loss of process component02 will now end the run, since it's a
BoardReader and there are no special rule(s) for it in the file
$DAQINTERFACE_PROCESS_REQUIREMENTS_LIST (if the file exists)
```

could possibly be re-worded to something like

```
Error: Ending run because BoardReader component02 has been lost.
This is the default action because there are no special rule(s) for it
in the file pointed to by $DAQINTERFACE_PROCESS_REQUIREMENTS_LIST.
(See user_sourcefile and/or docs/process_requirements_list_example)
```