DUNE DAQ Monitoring

Update and good practices

Marco Roda

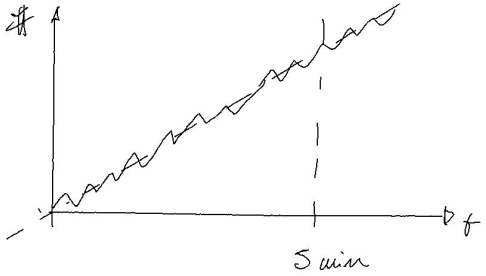
Overview

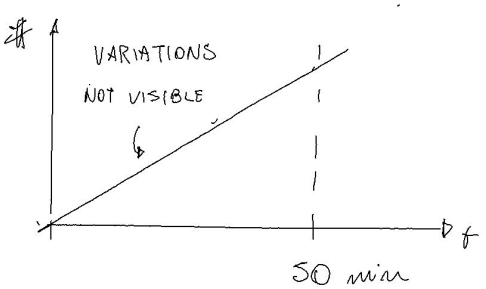
- Josh at the coordination meeting last week asked a few more details on how to write a metric
  - I put together something based on the experience so far
    - https://github.com/DUNE-DAQ/opmonlib/wiki/Good-practises-for-implementing-metrics
  - We have the instructions on how to create a schema and implement the metrics in the DAQModule code
    - They might not be perfect, but they are a good starting point
    - If you have questions, reach out to us
  - We have instructions on how to use the grafana instance
- Hopefully these good practices can connect the two instructions and give an bird eye view

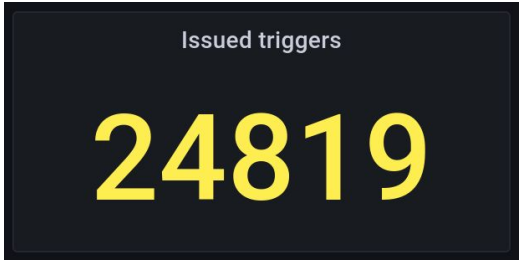
- In terms of concrete things to do
  - No need for particular CCM privileges to write your own metric
    - Just being able to push on the dedicated repository is enough
  - Once the metrics are implemented in the C++ code you basically have two options
    - Contact someone in the CCM to help create your dashboard
      - Most likely me
    - Prepare your own draft of a dashboard (you need an account for the instance we have on srv-009)
      - Then someone from CCM will integrate it for you - there might be some iterations

Proposed good practices

- Keep it simple
  - High level code can do complicated transformations
  - Simple metrics are easier to understand for a user
    - It will keep the dashboards intuitive
- Prefer quantities that are related to a single cycle of get_info() call.
  - Avoid run related values
    - typical example cumulative counters
  - Data are uncorrelated and easier to combine
  - Exceptions
    - Errors counters can be left run-related as having the information at the end of the run of the total errors is probably more useful than the frequencies of the error
    - Numbers you want to publish as numbers
- Invariance under frequency of the get_info() cycle or the collection interval
  - The dashboard rendering should report an information in a way which is independent from how often get_into() is called
  - or how many get_info() calls are used to construct a point in the final plot







Proposed good practices (continue)

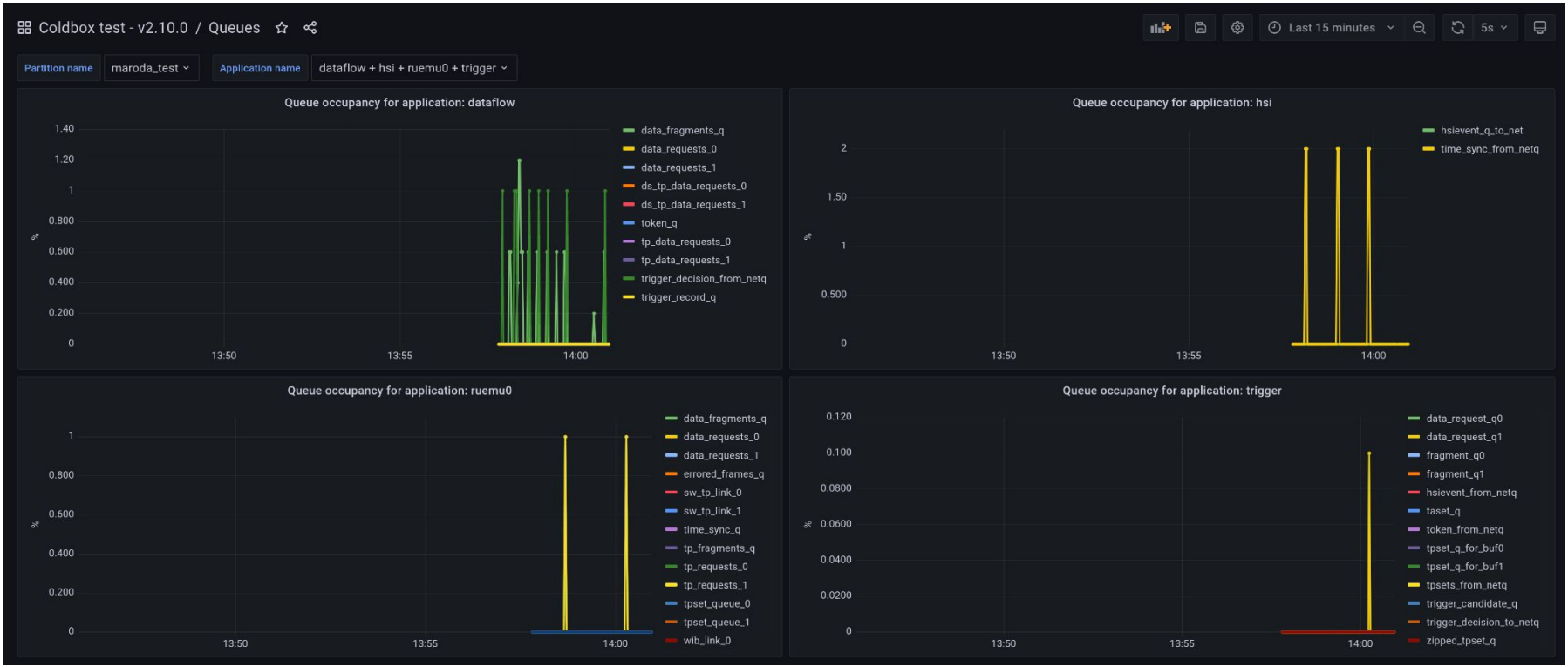
- Consider how the metric will scale according to configuration.
  - Most likely, every DAQModule is used more than once in a single DAQ partition
    - Each module constructs its own metric and eventually the dashboard will aggregate them together
      - it's possible to display each single value from each module, but it will overwhelm the user
      - When deciding which metric to implement to monitor a specific aspect of your DAQModule, please consider an effective aggregative plot that can display the relevant information and design your metric accordingly

  - Sometimes configuration swaps a DAQModule for another with a similar role but different underlying operations.
    - The standard paradigm for creating a schema suggests that every module should have it's own metric schema.
      - This require changes in the dashboards depending on the module.
    - But this is not necessary: each module can have multiple (factorised) schema and these schema should be re-used where appropriate across modules
    - Solution: identify common metrics that can be generated by the equivalent modules and use a single schema to publish all of them, in this way the dashboard changes will be limited.
      - Have a single module to have more than one schema associated with them, so unique information can be added as part of a different schema
      - Examples where this could be used already:
        - HSI modules (to be fixed because now it’s too late)
        - Timing monitoring: There should be a common timing info block that every timing endpoint has, including the timing system itself
          - In addition an hierarchy of monitorables, some of which are called often and some which are more detailed and are polled less often.

Proposed good practices (continue)

- Of course the main guide should be common sense and specific desirables or requirements from each module
  - Use these ideas to produce something homogeneous

New dashboard - Queue monitoring



Queue monitoring - some details

- It just plots the occupancies of each queue in the system
  - Divided by application

- The Application name variable allows the plots to be present in the dashboard that scales dynamically

Future plans

- Timing monitoring and DQM are going to be the next items
- The incoming NetworkManager
  - Development in the communications between DAQ apps
    - at the time with no specific monitoring
  - Adding a monitoring to the system is doable: I already had a discussion with Eric, main developer of NetworkManager
  - We will need to change Application to interrogate the NetworkManager
    - In a similar way as we changed Application to interrogate QueueManager
  - I haven't looked into the details yet, but there might be changes in the influxopmon as well
    - Alessandro and Alex should hear from me on this topic as soon as the code stabilises a bit
- Try to integrate more run control with monitoring to optimise the displayed information
  - First we need to work on RC itself
- Continue the support for the coldbox tests and all the new systems