DUNE DAQ Dashboards

A proposal

Marco Roda

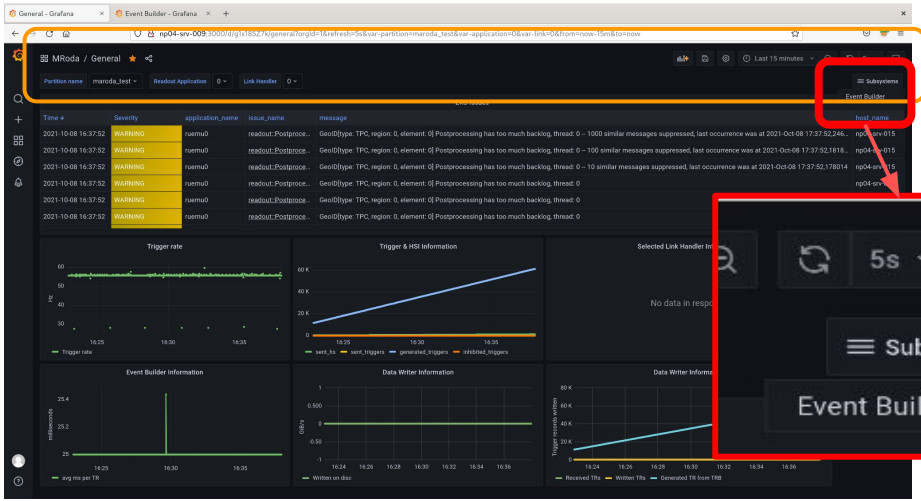
Overview

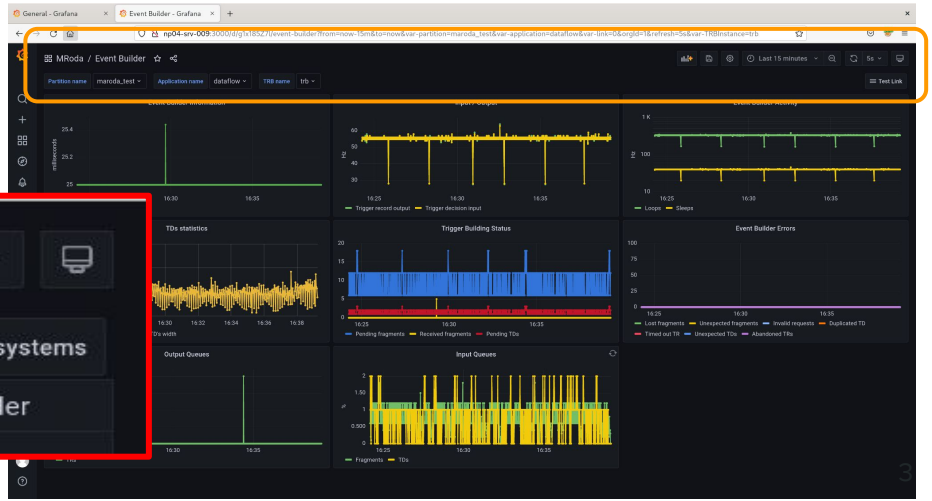
- We have a dashboard repo
  - Thanks to Alessandro for creating the repo
- Now the questions are:
  - How do we populate the repo?
    - which plots? standard/preferred formats
  - Who populates them?
  - What are the good practices?
    - Nice features we should use that are offered from Grafana
- I'm going to make a proposal
  - It's mostly the summary of what I found useful during debugging
  - Maybe I should say proposal**s**
    - I'm going to suggest many
  - These are just hints at the moment and not a full set of proposal
    - We can start from here and see if we like the suggestions
- I added a final section with open problems more specific for the coldbox test

Proposal for the graphics layout

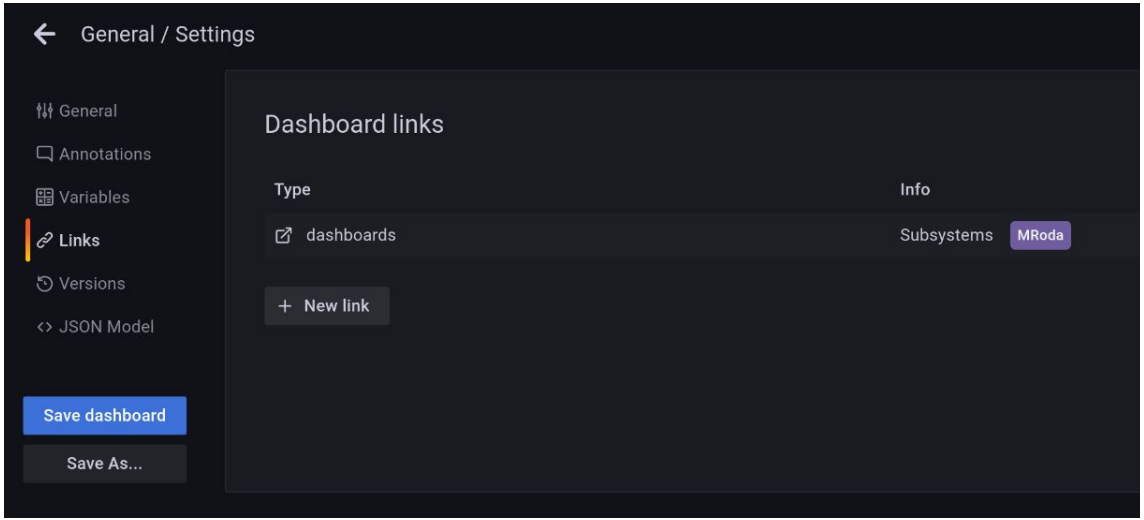
- The system is already too complicated to have all information on a single panel
  - Naively the idea is to use a general overview panel that can grant access to specific subsystem panels
  - Subsystem panels are not necessary related to physical subsystem
    - we can envision mixed panel if there is some informations that need to be cross checked simultaneously

- This can be achieved using “links” - Documentation





Links - and tags



- We can link anything
  - Or make a selection based on the tags of the dashboards
    - We can optimise our tags and links to obtain the desired behaviour

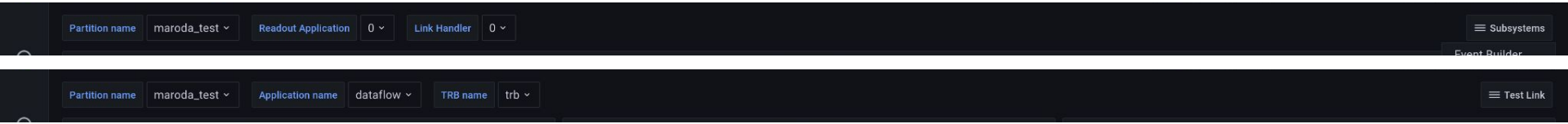
- Proposal
  - Subsystem dashboards can have the tag “Sharable”
    - And the mains can link against all the object tagged as Sharable
    - In this way we create a consistent set of dashboards
  - Subsystem can link against other dashboards as well
    - The presence of Sharable tag on the sub-sub dashboards will control if the lowest level dashboard will be linked in the main dashboard or not
  - We could have different links for subsystem that are application dependent (Event Building) and systems which are beyond partitions (Timing)

Possible usage cases of links

- Main
  - Subsystems in partition
    - Event Building
    - Trigger(s)
    - Readout
      - ....
    - DQM
  - Resources
    - Timing
    - network
    - Disk usage
    - CPU/RAM
    - Servers (same information but different aggregation)

Variables

- Variables are extremely useful to allow different users to use different dashboards
  - Archetypical example is the partition name
  - application and module names are probably as important in our DAQ system
- Variables are ported in linked dashboards when opened
- Proposal
  - Dashboards should contain variables as much as possible
  - variables to be inherited from the main (or in general from upper level dashboard) should have the same name
    - Variables that should not be inherited should have different names



Proposal for some good practices

- The basic assumption is
  - “Module developers should know best the behaviour of their code”
  - They should be able to understand the system to provide the best dashboard for a certain subsystem
    - Some metrics are not intuitive or even just complicated
  - Of course with feedback from users

- Proposal
  - Module developers should prepare dashboard(s) for their subsystem
  - For mixed dashboards, we should have someone working on them
    - Example: connection monitoring
  - Make sure that the rendering of the metrics on the dashboards should not be sensitive to the OPMON interval
    - Otherwise users will be confused

Open questions

- This ties a bit with the metrics implemented by the developers:
  - Shall we have common guidelines?
    - E.g. counter metrics - Is it worth having counters for the whole run, for time interval, etc when these can be constructed via the dashboards?
  - This might make things easier for the developers

- What is the most readable format?
  - Counters? frequencies?
  - Feedback from user needed
    - Hopefully this will be more clear after the coldbox tests

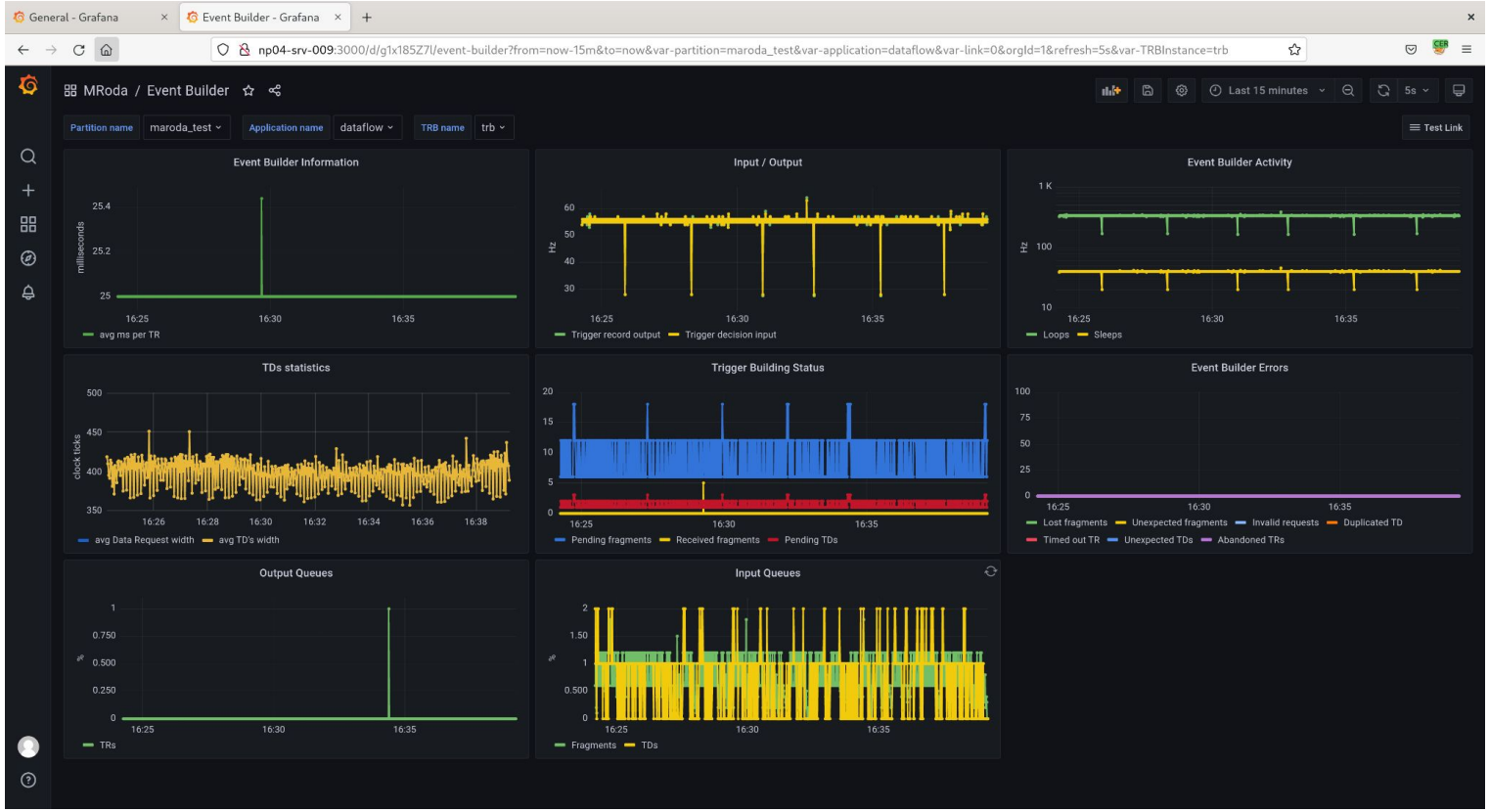
Possible next step

- If we like these ideas we can
  - formalise these guidelines in the README of grafana-dashboards
    - Are there better entry points for this kind of documentation?
- Start producing an inventory of the available dashboards
  - For sure not all subsystems have a dedicated “sherable” dashboard
    - Even if there is not much to show for a single dashboard having a starting point is useful for future changes
    - It will help the creation of main
  - Take actions for the missing one
- Start working on a possible main
  - I think the repo already contains a good one, we can add links, simplify it a bit
- Is it possible to have pockets so that it takes the latest version of dashboards?
  - We should consider starting adding tags to the dashboard repository as well
  - To make the dashboards part of the releases

Preparation for Coldbox

Some open questions to share and cross-check

- We need to specify which Grafana instance(s) will we use for looking at OpMon metrics from the Coldboxes, VST, and NP02/SSP
  - My guess is that it will be np04-srv-009
  - Even if the tester will use pocket instances for the minidaq app, they can send data to the np04-srv-009 grafana instance
- What about a standard or recommended dashboards for the opmon metrics?
  - What will the names of those dashboards be?
    - If we follow the previous plan, we just need to deploy initial dashboard and the others will follow through
  - What is the documentation needed by the coldbox testers?
    - Hopefully once we have this information we can prepare a page for them that contains all this information
- How many different test environment we need to support simultaneously?
  - Will it be enough to change the partition name in the dashboard to support them all?
    - My guess is yes, but I'm happy to take feedback
- What dashboard that are needed for the coldbox test are simply completely missing?
  - Is it a problem of the dashboard only or the metrics themselves have to be written?



That’s all folks