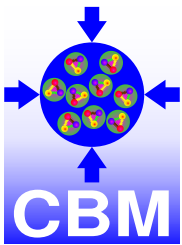
A data Quality-Assurance framework for online and offline applications for the CBM experiment

CHEP2024, Kraków, Poland

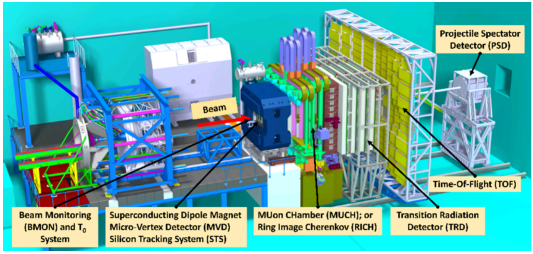


Sergei Zharko
for the CBM Collaboration

CBM Department
GSI Helmholtzzentrum für Schwerionenforschung
Darmstadt, Germany

October 22, 2024

The Compressed Baryonic Matter experiment

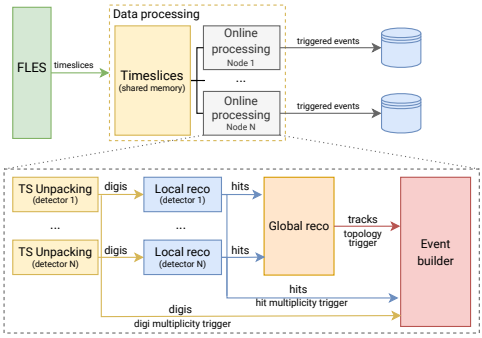


CBM@FAIR (Darmstadt, Germany):

- exploration of QCD phase diagram at large μ_B
- fixed-target setup, under construction (available from 2028, test mini-CBM setup takes data)
- high interaction rates – 10 MHz
- free-streaming readout, software event triggering

Online data processing:

- triggering and storing interesting events
- full event reconstruction
- reduction of data rates from 1TB/s to 10GB/s



Data quality assurance in CBM

QA-task(module) – fixed set of reference histograms:

- detector subsystem calibration and occupancy
- reconstruction algorithms performance

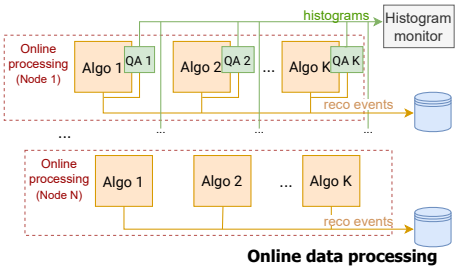
⇒ **data-driven automatic and systematic bug/malfunction monitoring**

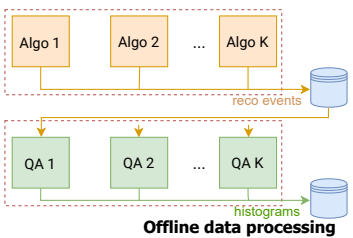
Online scenario:

- processing data from each reco algorithm for **selected** nodes
- **performance-critical** ⇒ limited number of histograms
- monitoring (no analysis step)

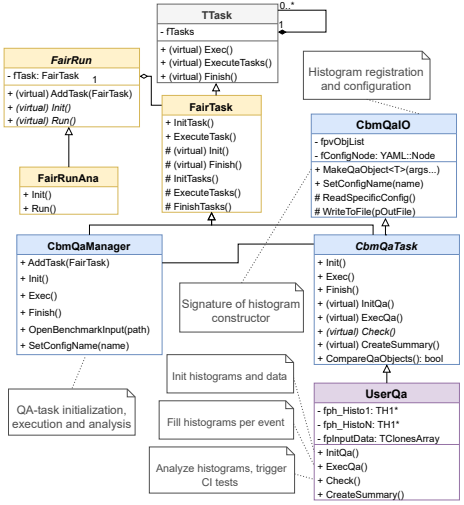
Offline scenario:

- post-processing of stored reco events
- available for simulated data as well
- analysis step: data-driven problem detection and validation (CI)



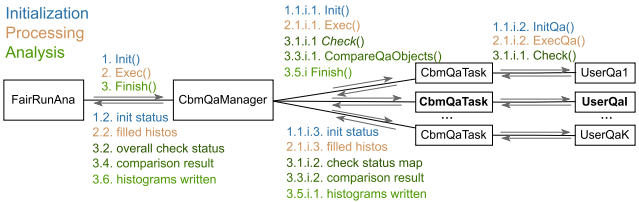


Offline scenario for data QA



Framework:

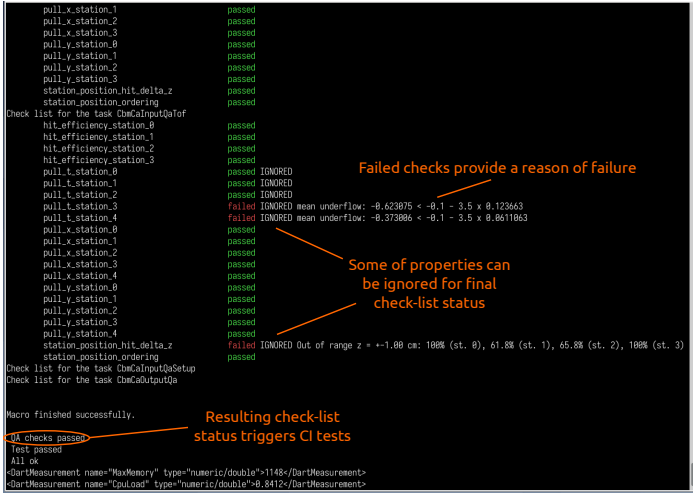
- a set of histograms, filling approach, validation criteria of histogram properties from user
- tasks are processed sequentially on a single thread
- all details are unified and automatised



Analysis step:

- check of histogram properties, requested by user
- compare of current histograms with the benchmark ones (different code version, geometry, ...)

Offline scenario for data QA



QA check-list:

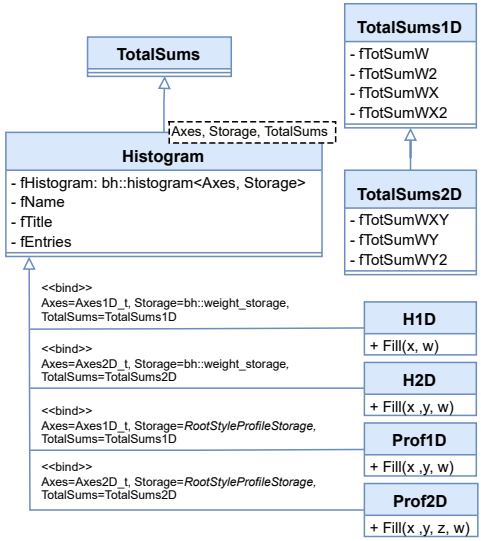
- data-driven checks: properties of collected histograms
- multiple checks for each QA-task
- failed checks are collected by the QA-manager and trigger CI-tests

Online scenario for data QA

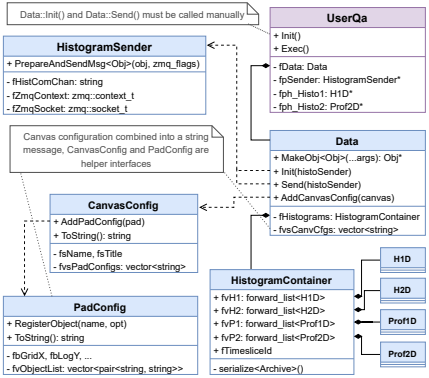
Avoiding ROOT dependencies on the data processing nodes for performance

⇒ **custom histograms**:

- based on `Boost::histogram`
- ROOT-like basic interface (ctors, fill, reset)
- fully reproduce statistical properties (errors, moments)



Online scenario for data QA

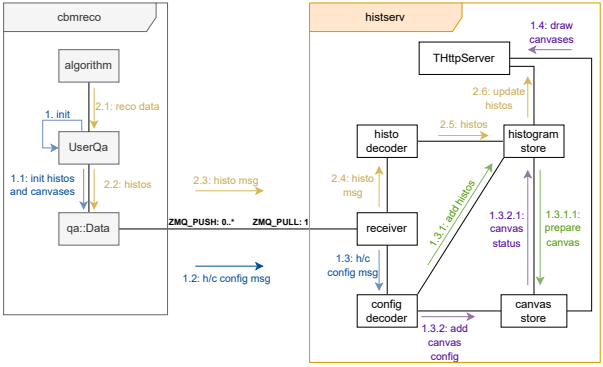


QA modules:

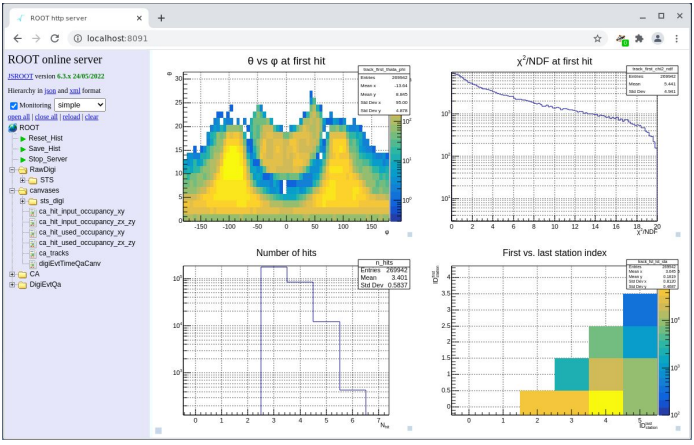
- keep histograms on stack in lists
- hide details from user
- send histograms to the histogram monitor via ZMQ_PUSH socket after each processed TS

Histogram monitor (histserv):

- standalone process, has ROOT
- receives histograms from multiple reco nodes via ZMQ_PULL socket (**a single socket both for configuration and content**)
- no information on source QA-modules
- updates histograms to the web-interface



Online scenario for data QA



Histogram web-interface:

- ROOT `THttpServer`
- shows/stores integrated histograms and(or) histograms vs. timeslice ID

Summary

- Flexible and user-friendly data QA framework is implemented and used in the CBM experiment:
  - offline QA is a part of official CI tests as a machinery for problem catching
  - online QA performed successfully in mini-CBM runs in spring of 2024

- Several features under development:
  - version control: histogram comparison in different setups for different code version
  - configurable histogram property checks in the online scenario (YAML and UI)
  - combining of the online and offline QA-frameworks (a single UserQA class for both scenarios)

Thank you for your attention!