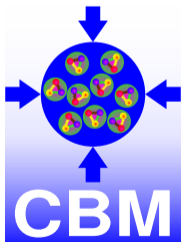


# A data Quality-Assurance framework for online and offline applications for the CBM experiment

CHEP2024, Kraków, Poland



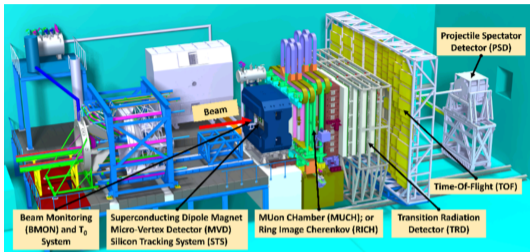
Sergei Zharko  
for the CBM Collaboration

CBM Department  
GSI Helmholtzzentrum für Schwerionenforschung  
Darmstadt, Germany

October 22, 2024



# The Compressed Baryonic Matter experiment

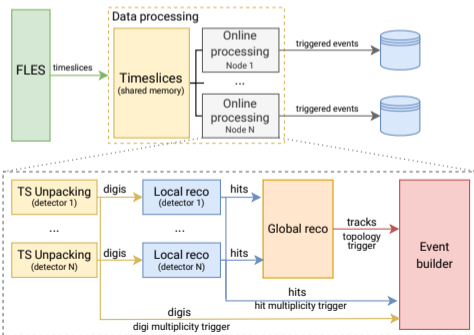


## CBM@FAIR (Darmstadt, Germany):

- exploration of QCD phase diagram at large  $\mu_B$
- 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

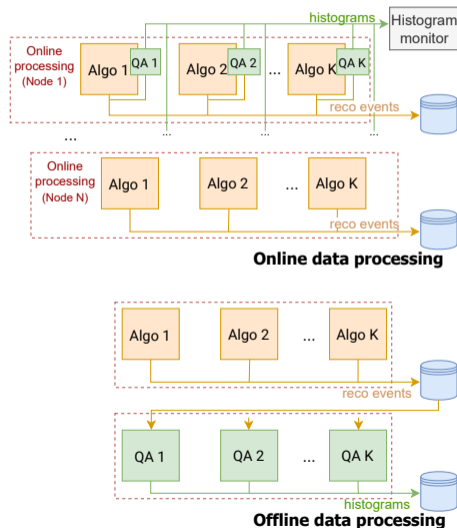
⇒ **data-driven automatic and systematic bug/malfunction monitoring**

**Online scenario:**

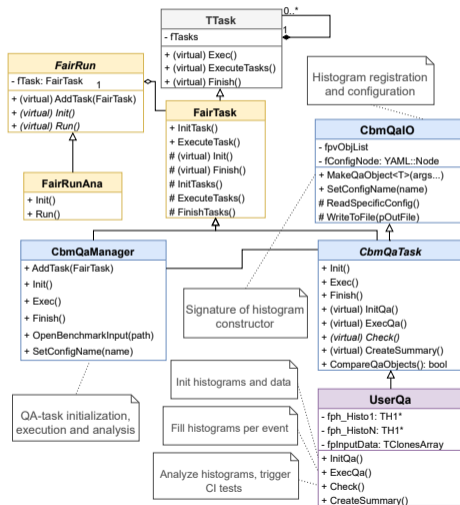
- processing data from each reco algorithm for **selected** nodes
- **performance-critical** ⇒ limited number of histograms
- monitoring (no analysis step)

**Offline scenario:**

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



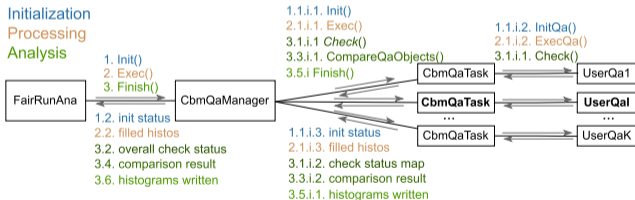
# Offline scenario for data QA



## 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 automatized

## Initialization Processing Analysis



## 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

```
pull_x_station_1      passed
pull_x_station_2      passed
pull_x_station_3      passed
pull_y_station_0      passed
pull_y_station_1      passed
pull_y_station_2      passed
pull_y_station_3      passed
station_position_hit_delta_z  passed
station_position_ordering  passed
Check list for the task CbmCaInputQaToF
hit_efficiency_station_0  passed
hit_efficiency_station_1  passed
hit_efficiency_station_2  passed
hit_efficiency_station_3  passed
pull_t_station_0        passed IGNORED
pull_t_station_1        passed IGNORED
pull_t_station_2        passed IGNORED
pull_t_station_3        failed IGNORED mean underflow: -0.623075 < -0.1 - 3.5 x 0.123663
pull_t_station_4        failed IGNORED mean underflow: -0.373006 < -0.1 - 3.5 x 0.0611063
pull_x_station_0        passed
pull_x_station_1        passed
pull_x_station_2        passed
pull_x_station_3        passed
pull_x_station_4        passed
pull_y_station_0        passed
pull_y_station_1        passed
pull_y_station_2        passed
pull_y_station_3        passed
pull_y_station_4        passed
station_position_hit_delta_z  failed IGNORED Out of range z = +-1.00 cm: 100% (st. 0), 61.8% (st. 1), 65.8% (st. 2), 100% (st. 3)
station_position_ordering  passed
Check list for the task CbmCaInputQaSetup
Check list for the task CbmCaOutputQa

Macro finished successfully.
QA checks passed
Test passed
All ok
<DartMeasurement name="MaxMemory" type="numeric/double">1148</DartMeasurement>
<DartMeasurement name="CpuLoad" type="numeric/double">0.8412</DartMeasurement>
```

Failed checks provide a reason of failure

Some of properties can be ignored for final check-list status

Resulting check-list status triggers CI tests

## QA check-list:

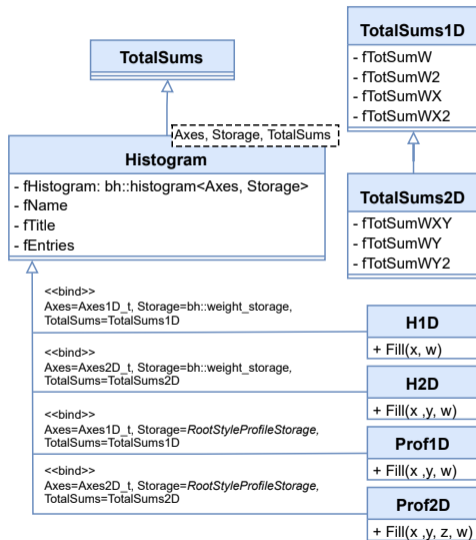
- data-driven checks: properties of collected histograms
- multiple checks for each QA-task
- failed checks are collected by the QA-manager and trigger CI-tests

# Online scenario for data QA

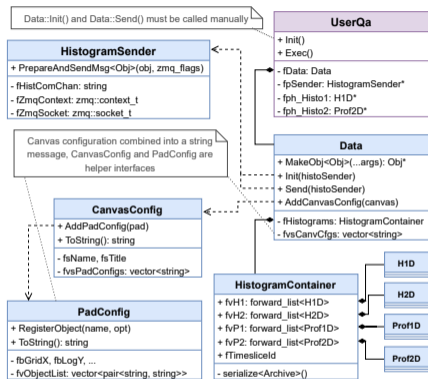
Avoiding ROOT dependencies on the data processing nodes for performance

⇒ **custom histograms:**

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



# Online scenario for data QA

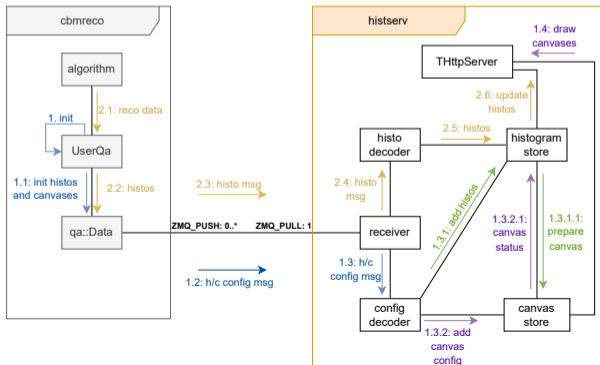


## QA modules:

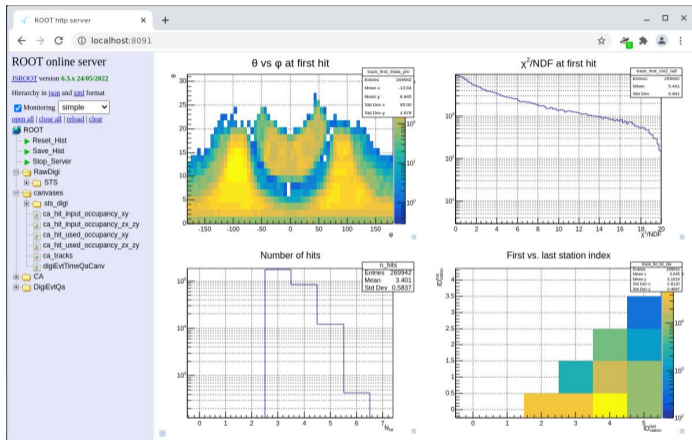
- keep histograms on stack in lists
- hide details from user
- send histograms to the histogram monitor via ZMQ\_PUSH socket after each processed TS

## 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!**