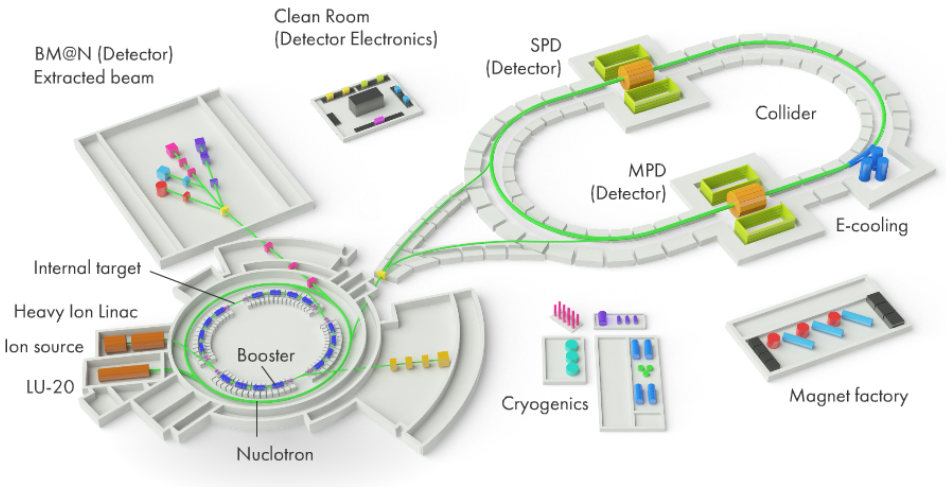
Online processing and QA of the BM@N experiment

Ilnur Gabdrakhmanov

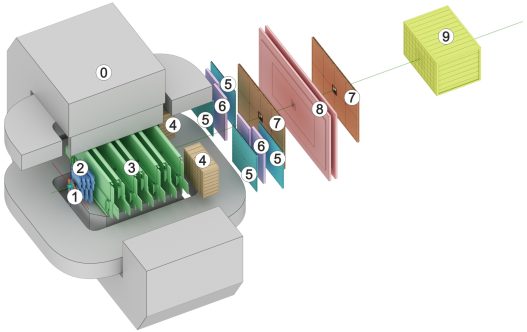
Joint Institute for Nuclear Research, Laboratory of High Energy Physics

The XV-th International School-Conference "The Actual Problems of Microworld Physics
Westa
Aug 30, 2023

Nuclotron based Ion Collider fAcility complex



Baryonic Matter at Nuclotron

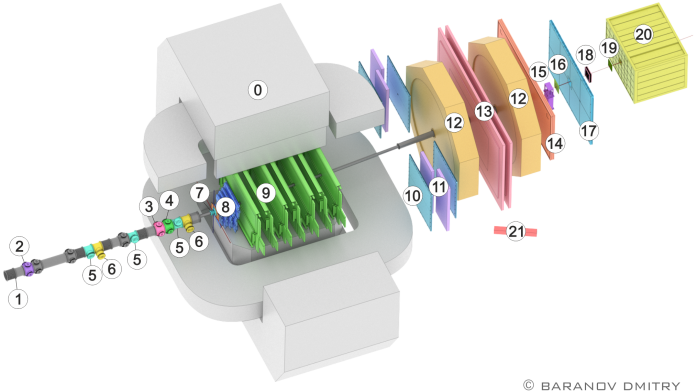


- Magnet SP-41 (0)
- Triggers: BD + SiD (1)
- Forward Silicon (2)
- GEM (3)
- ECAL (4)
- CSC 1x1 m^2 (5)
- TOF 400 (6)
- CSC 2x1.5 m^2 (7)
- TOF 700 (8)
- ZDC (9)

Subsystems

- Trigger system:
  - Beam counters
  - Barrel detector
- Tracking system:
  - Forward Silicon
  - GEM (Gas Electron Multipliers)
  - CSC (Cathode Strip Chambers)
- Identification(time-of-flight) system:
  - ToF-400
  - ToF-700
- Calorimeters:
  - ZDC (Zero Degree Calorimeter)
  - ECAL (Electromagnetic Calorimeter)

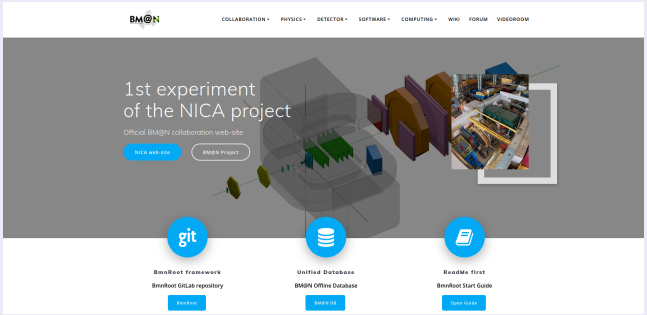
Baryonic Matter at Nuclotron



- Magnet SP-41 (0)
- Vacuum Beam Pipe (1)
- BC1, VC, BC2 (2-4)
- SiBT, SiProf (5, 6)
- Triggers: BD + SiMD (7)
- FSD, GEM (8, 9)
- CSC 1x1 m^2 (10)
- TOF 400 (11)
- DCH (12)
- TOF 700 (13)
- ScWall (14)
- FD (15)
- Small GEM (16)
- CSC 2x1.5 m^2 (17)
- Beam Profilometer (18)
- FQH (19)
- FHCal (20)
- HGN (21)

BM@N

BM@N Framework BMNROOT



BM@N experiment home web-page:

https://bmn.jinr.ru

- News
- Software repositories
- Software tests
- Forums
- Database for physics run
- E.t.c.

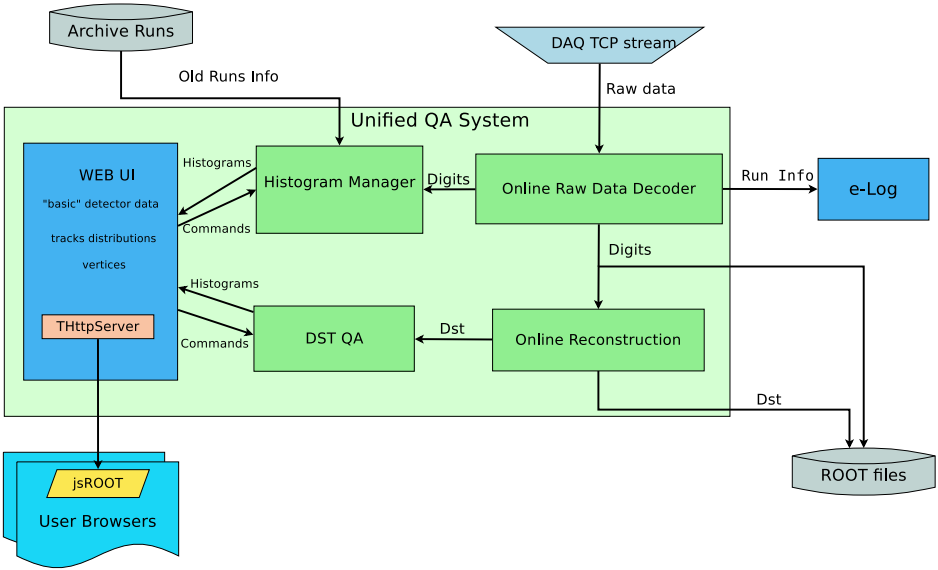
Benefits:

- Inherits basic properties from FairRoot (https://fairroot.gsi.de/), C++ classes
- Detector composition and geometry; particle propagation by GEANT3/4
- Advanced detector response functions, realistic tracking and PID included
- Event display for Monte-Carlo and experimental data
- QA system

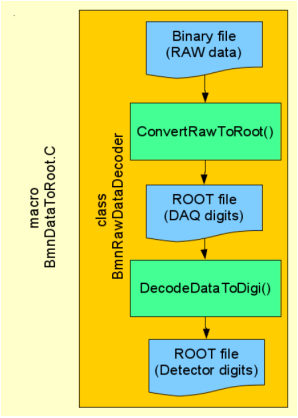
BmnROOT repository

https://git.jinr.ru/nica/bmnroot

General system scheme



Decoding scheme



First step (Data Converter):

- Read a binary data file with RAW-data.
- Parse the data blocks: run/spill/event/module.
- Create «DAQ-digits» (ADC, TDC, TQDC, HRB, SYNC, etc.) accordingly DAQ-data-format and write them into a tree.

Second step (Data Decoder):

- Read detector mappings (channel-to-strip) from the Unified Database
- Calculate pedestals and common modes of channels
- Clear noisy channels
- Decode DAQ-digits into detector-digits (BmnGemDigit, BmnTofDigit, etc.)
- Write the tree with detector-digits to a ROOT-file

Basic QA frontend with hardcoded histograms

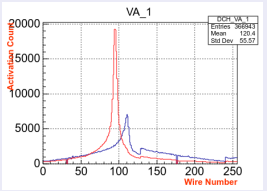
Implementation details:

- The data processed and transferred from the previous stage is used to fill ROOT histograms. Which in turn are sent to the end users via http.
- CERN jsROOT library is used to transform the ROOT object to the html histograms.
- Base class for histogram sets BmnHist is used in:
  - BmnHistTrigger
  - BmnHistGem
  - BmnHistToF
  - ...

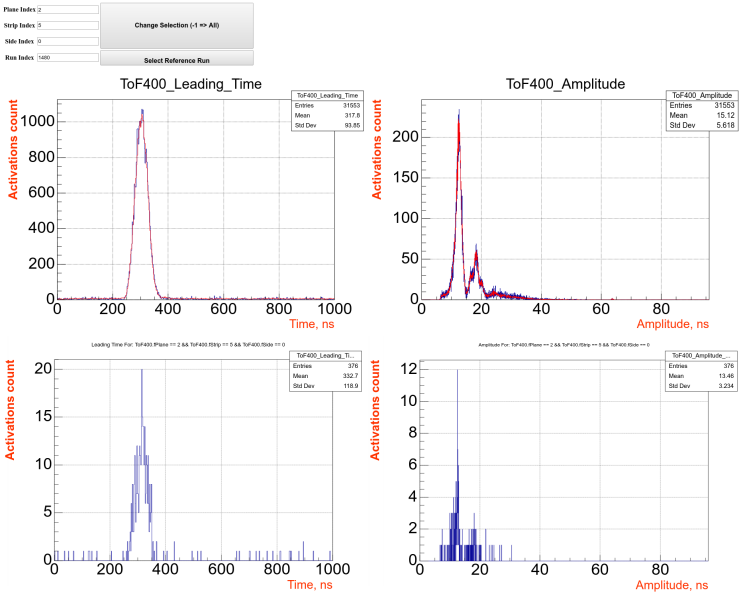
Thus addition of the new detector histogram set is rather simple.

Reference run:

- ✓ Ref run imposition
- ✓ Autoselection of similar runs



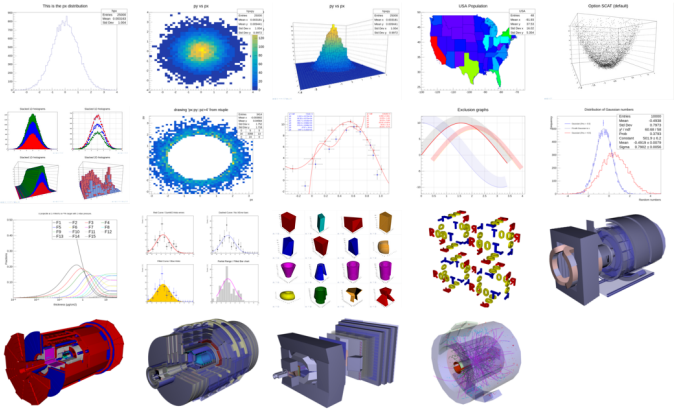
Fine grain selection (station/plane/strip):



jsROOT

CERN jsROOT library:

ROOT object
⇓
HTML visual object



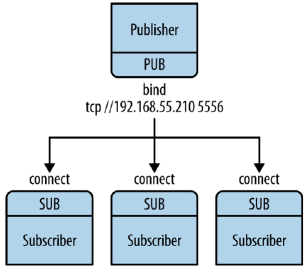
jsROOT website

https://root.cern.ch/js/

ØMQ

Lightweight high-speed library for network messaging

- Variety of transports: TCP, interprocess, inroc
- Automatic queue and buffer managing
- Many usable messaging patterns



ZeroMQ website

https://zeromq.org

Existing alternative online processing frameworks

- TDAQ (ATLAS)
  - tightly integrated with other ATLAS software
  - thus it is rather difficult to deploy in other program environment
- FairMQ (GSI FAIR) (I.Romanov, K.Gertsenberger are working on applying it to bmnroot)
  - seems to be quite flexible in deployment and settings (with DDS as an option)
  - but requires additional wrapper code
  - seems not to work in an interactive ROOT macros

FairRoot way of analysis via FairTask's
(Extensively being used in the BmnRoot)

- FairRunAna - task manager class
- FairSource - abstract class for a data source
- FairSink - abstract class for a data destination manager

Typical analysis macro workflow:

- BmnFileSource/FairFileSource (input data file)
- Task1 (executed event-by-event)
- Task2
- Task3
- ...
- FairRootFileSink (output data file)

Simplest way to move existing reconstruction code to online

Less code $\rightarrowtail$ Less errors

ZMQ transfer classes for FairRunAna

- BmnMQSource - ZeroMQ SUB socket[1] based source class
- BmnMQSink - ZeroMQ PUB socket based sink class

Benefits

- No need to rewrite existing bmnroot analysis code. (No need to touch any working task)
- It became possible to combine several analysis macros by source/sink network interfaces

[1] https://zeromq.org

BmnRoot QA structure

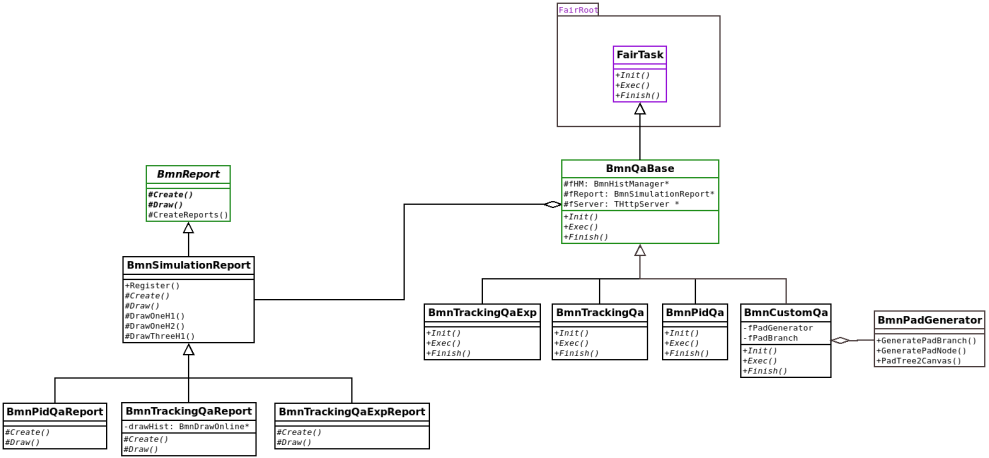
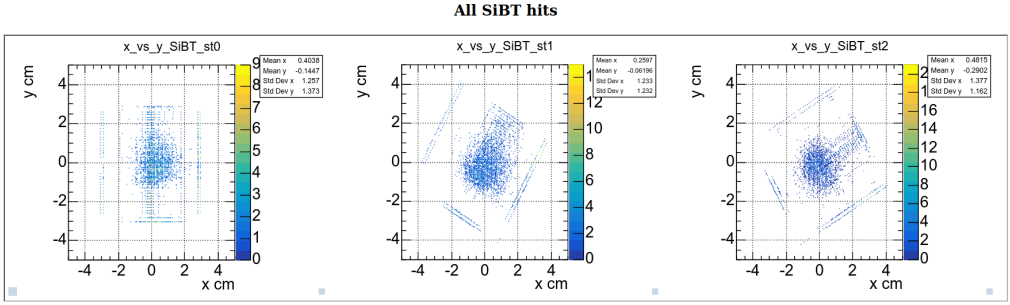
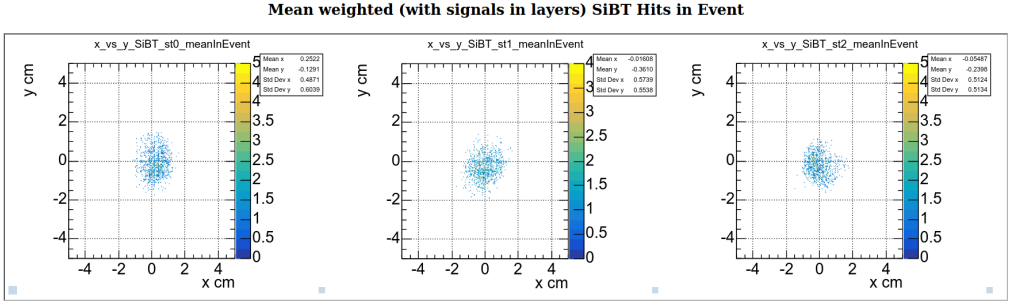


Figure: QA main classes (green ones were forked from CbmRoot)

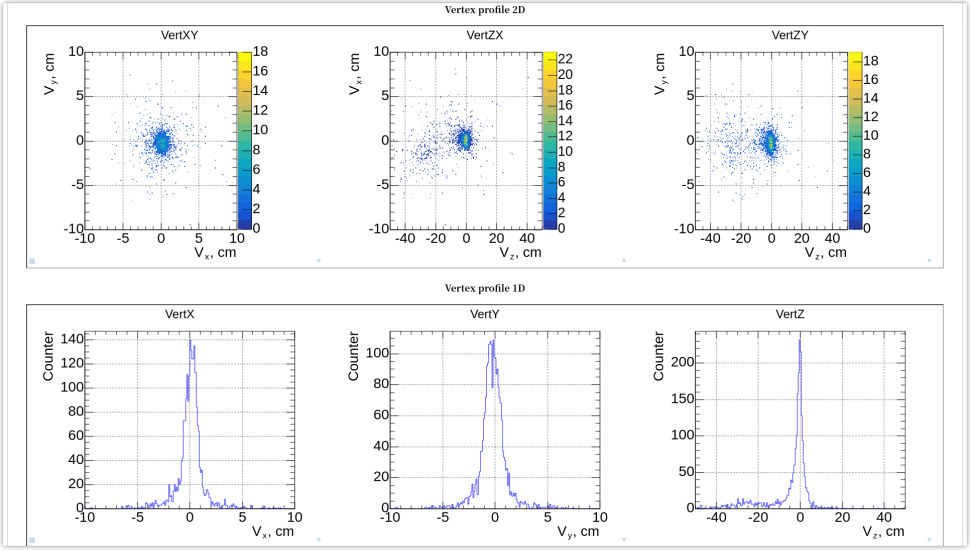
Live example of the online hit reconstruction & QA (SiBT as the example)



Live example of the online hit reconstruction & QA (SiBT as the example)



Live example of the primary vertex online reconstruction



Custom «no code» histograms. Motivation

Why?

Experiment upgrade as well as conduction of two experimental setups require distribution of work on the development of the online QA system.
Namely each detector team should be able to extend system's functionality easily.

Custom «no code» histograms. Motivation

Why?

Experiment upgrade as well as conduction of two experimental setups require distribution of work on the development of the online QA system.
Namely each detector team should be able to extend system's functionality easily.

Main objectives:

- Move monitoring configuration outside of the code
- Make addition of histogram simple and flexible (It should not require code rebuild)
- Implement filling logic configurable as well (thanks to ROOT TTree::Draw text parser it was possible)

Custom «no code» histograms. Motivation

Why?

Experiment upgrade as well as conduction of two experimental setups require distribution of work on the development of the online QA system.
Namely each detector team should be able to extend system's functionality easily.

Main objectives:

- Move monitoring configuration outside of the code
- Make addition of histogram simple and flexible (It should not require code rebuild)
- Implement filling logic configurable as well (thanks to ROOT TTree::Draw text parser it was possible)

Implementation

BmnPadGenerator class - creates a pad structure in the canvas on the basis of json scheme.

Test code example:

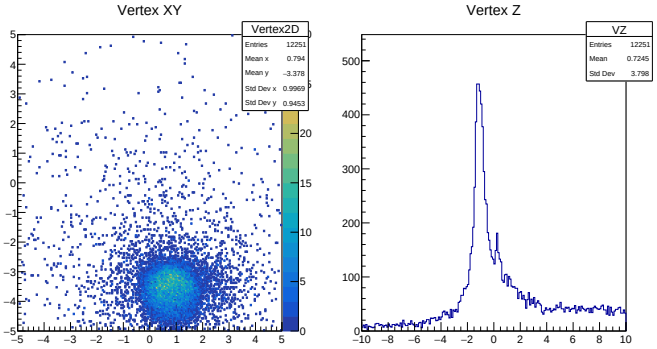
```
BmnPadGenerator *g = new BmnPadGenerator();
g->LoadPTFrom(FileName);
BmnPadBranch * br = g->GetPadBranch();
TCanvas* can = new TCanvas("canHits", "", 1920, 1080);
g->PadTree2Canvas(br, can);
BmnHist::DrawPadTree(br);
```

Simple configuration

JSON scheme:

```
{
    "Name": "Custom canvas",
    "Title": "Custom Canvas",
    "DivX": "2",
    "DivY": "1",
    "Pads": [
        {
            "Class": "TH2F",
            "Name": "Vertex2D",
            "Title": "Vertex XY",
            "Variable": "BmnVertex.fY:BmnVertex.fX",
            "Selection": "(BmnVertex.fZ>-10 && BmnVertex.fZ<10)",
            "Options": "colz",
            "Dimensions": [
                200,
                -5,
                5,
                200,
                -5,
                5
            ]
        },
        {
            "Class": "TH1F",
            "Name": "VZ",
            "Title": "Vertex Z",
            "Variable": "BmnVertex.fZ",
            "Selection": "(BmnVertex.fZ>-10 && BmnVertex.fZ<10)",
            "Dimensions": [
                200,
                -10,
                10
            ]
        }
    ]
}
```

Canvas structure:



- Works well for data in TClonesArray branches
- Doesn't work for single object branches out of the box (only with additional code for each class)
- User interface for scheme updating is not yet ready

QA experience overview. Possible future improvements.

- jsROOT:
  - Sometimes not updating scales in the jsROOT histograms. Even after restart of a QA.
    - Maybe we should move to Grafana, but it will require rewriting basic histograms functions
- DST QA:
  - Slow tracking procedure (even L1)
    - Actually not a problem (the character of distribution stays the same. ZMQ just drops events not fitting the buffer)
    - sometimes gives noticable lag of processing cached events of a previous run for a couple of minutes
  - Needs to fully implement reference runs in the unified QA as well:
    - Maybe we should use UniDB to store reference run sets
  - Needs to finish Custom QA (no code). Particularly implement user interface for schemes.
- Practice students helped with the work:
  - A. Islentev (The University Edinburgh) - Converter parallelization, Decoder denoising improvements and fixes
  - K. Mashitsin (SPbU) - GEM decoding algorithm improvements and fixes
  - A. Driuk (SPbU) - Digi correlation histograms, SiBT histograms
  - A. Iufriakova (SPbU) - ADC decoder SIMD optimisation

Conclusion

- Unified online/offline QA system is being developed in the framework of the bmnroot package
- Which also included optimization of the data decoding procedures
- ZeroMQ network transfer classes were developed for FairRunManager based analysis
- "No code" approach were developed in order to simplify extension of the system

Conclusion

- Unified online/offline QA system is being developed in the framework of the bmnroot package
- Which also included optimization of the data decoding procedures
- ZeroMQ network transfer classes were developed for FairRunManager based analysis
- "No code" approach were developed in order to simplify extension of the system

Thanks for your attention!