

# QA histograms for TPC online monitoring

ALEXANDER KRYLOV

DUBNA/JINR

EMAIL: AVKRYLOV@JINR.RU

# Event display and online monitoring

**Event display** focuses on visualizing individual events in detail, showing 3D TPC geometry and information after some reconstruction methods.

**Online monitoring** aims to oversee the overall performance of the experiment and the quality of data being collected in real time.



## Preliminary TPC monitoring scheme



### Quality assurance

- Inner pads ADC per sector 24 histograms
- Outer pads ADC per sector 24 histograms
- Inner pads ADC per timebucket 24 histograms (per each sector)
- Outer pads ADC per timebucket 24 histograms (per each sector)
- Inner pads ADC for current event 24 histograms (per each sector)
- Outer pads ADC for current event 24 histograms (per each sector)
- General clusters information 6 histograms

Total number of histograms > 150



Quality assurance from STAR experiment

### ADC distribution per sector (current event)

### NICA) MPD TPC - 😳 Socket 🗠 Slow Control 📰 About



### ADC distribution per sector (storage)

### 😑 🕋 🕂 🕅 🗚 TPC 🝷 🛟 Socket 🗠 Slow Control 🖬 About

Sec: 0 Charge per Pad Sec: 1 Charge per Pad ×10<sup>3</sup> ×10<sup>3</sup> 40 40 35 20 15 10 10 40Ē 18 35 30 25 10 15 0E -60 -40 40 60 Pads -60 -40 40 60 Pads -20 0 -20 20 Sec: 2 Charge per Pad Sec: 3 Charge per Pad ×10³ ×10<sup>3</sup> ChargePadRow\_3 163519 Entries 22 -1.062 Mean x 45 40 35 20 15 10 10 20 Mean v 26.94 Std Dev x 23.12 40 35 30 25 20 15 10 18 14.75 Std Dev v 60 Pads -40 -40 40 60 Pads Sec: 4 Charge per Pad Sec: 5 Charge per Pad ChargePadRow\_5 ChargePadRow\_4 155829 171147 Entries Entries 30 50 L § 50 Mean x 0.7755 Mean x -0.4724 26.93 26.36 Mean y Mean y

**QA** Controls QA type General Histograms Charge per PadRow (Run) Charge per TimeBin (Run) Charge per Pad (current, event № -1) Pad layout Inner Pads Outer Pads Cuts Charge 0 10% 40% 50% 60% 70% 80% 90% 100 20% 30% Phi 20% 30% 40% 50% 60% 70% 80% 90% 100 0 10% Theta 0 3 4 5 6 7 10

### Event ID: 0 🔀 🕨 🗶

### ADC distribution per time (Inner pads)



70%

70%

7

6

80%

80%

8

90%

90%

0

100

10

### ADC distribution per time (Outer pads)



**QA** Controls QA type General Histograms Charge per PadRow (Run) Charge per TimeBin (Run) Charge per Pad (current, event № -1) Pad layout Inner Pads Outer Pads Cuts Charge 0 10% 30% 40% 50% 60% 70% 80% 90% 100 20% Phi 0 20% 30% 40% 50% 60% 70% 80% 90% 100 Theta 0 4 5 6 8 9 10









Helix

## Clusters info

### NICA MPD TPC - 🕼 Socket 🗠 Slow Control 🖿 About



10

10

# JSROOT library

The JSROOT library allows:

- reading of binary and JSON ROOT files in JavaScript/TypeScript;
- drawing of different ROOT classes in web browsers;
- reading and drawing TTree data;
- interacting with histograms and graphs
- using in node.js.



### JSROOT possibilities

### 😑 🕋 🕂 NICA MPD TPC 🔹 🕼 Socket 🗠 Slow Control 🖿 About

### Event ID: 0 🛛 🖌 🕨 🗙



# New Technologies

	New (ED + QA)	Old (Event Display)
FrontEnd	React + TypeScript	React + JavaScript
BackEnd	Bun	NodeJS
Server	Elysia	Express
Store	Modern Redux	Redux
Bundler	Vite	WebPack
Connection	WebSocket	WebSocket
Graphics	ThreeJS + JSROOT	ThreeJS

## TypeScript vs JavaScript



# Functional components (React)

**Simplicity**: They are just a function, which makes them simpler to write and understand. There's no need to worry about **this** keyword, which can confuse some developers.

**Performance**: Functional components can be slightly more efficient than class components. They avoid the overhead of class objects and are more accessible for React to optimize.

**Ease of Testing**: Functional components are generally easier to test since they are just functions without side effects.

**Conciseness**: They often result in less code, making your codebase cleaner and easier to maintain.

**Hooks!!!**: With the introduction of React Hooks, functional components can now manage state, side effects, and more, which were previously only possible in class components.

### Bun

**Bun** is a new JS runtime focused on performance and being all-in-one runtime, bundler, package manager and transpiler.

**Speed**. Bun starts fast and runs fast. It extends JavaScriptCore, the performance-minded JS engine built for Safari.

**APIs.** Bun provides a minimal set of highly-optimized APIs for performing common tasks, like starting an HTTP server and writing files.



### Conslusion

Online monitoring as well as event display are one of the most important tools to control quality of data during an experiment.

Its combination ensures that not only data collection process efficient and reliable, but the data itself is rich and informative.



# Thank you for your attention!

