Online TPC data monitoring for the MPD experiment

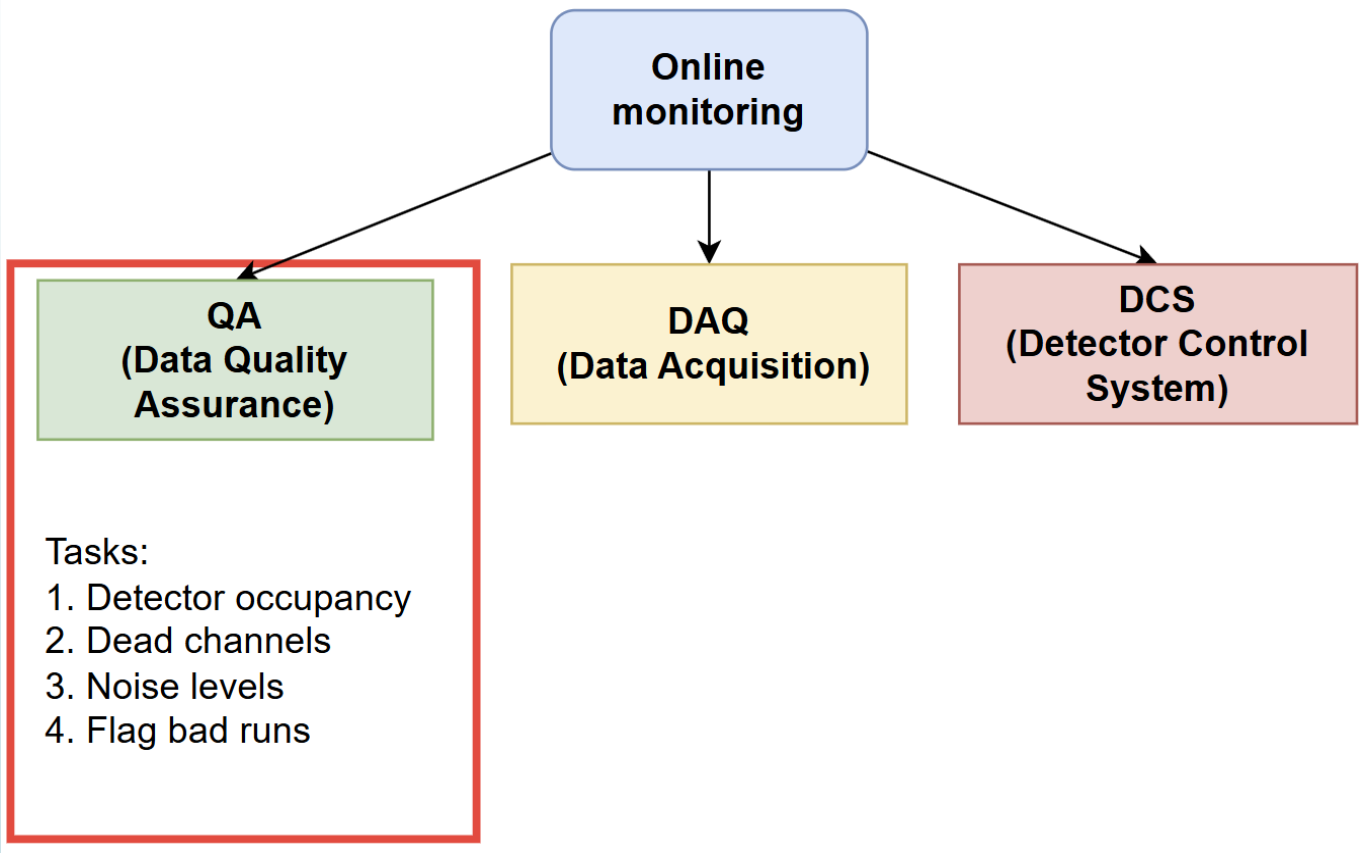
JINR/LHEP A.Krylov

Email: avkrylov@jinr.ru

Online monitoring tasks

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

QA main goal is to **reveal detector problems** and data-quality losses early enough for the shifter to react during the run.

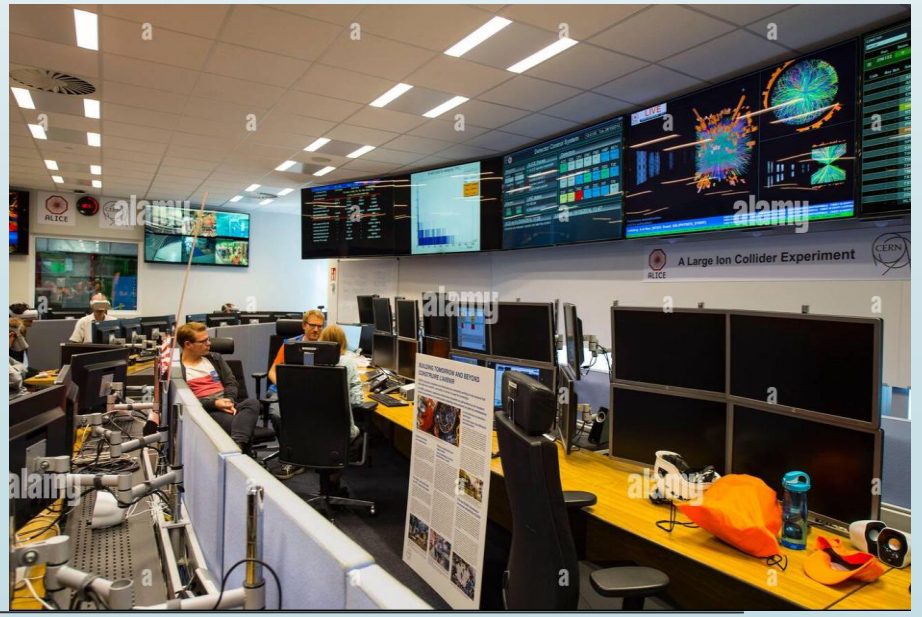


Control room

Control room is the operational center of the experiment where shifters supervise detector safety, data taking, and data quality through online monitoring.
Here, monitoring information is turned into real-time operational decisions.



ATLAS control room

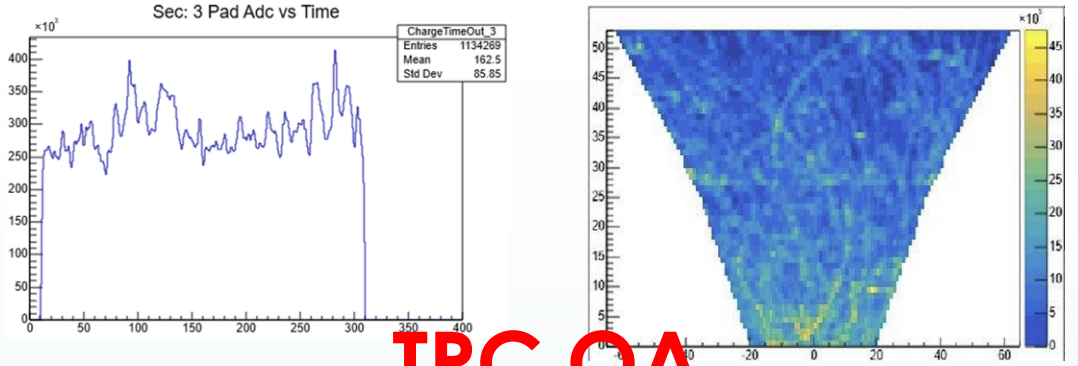


ALICE control room

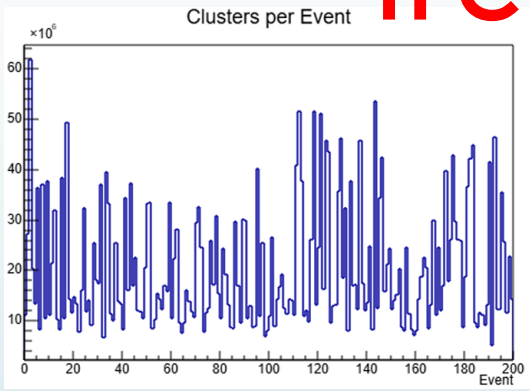
TPC data information

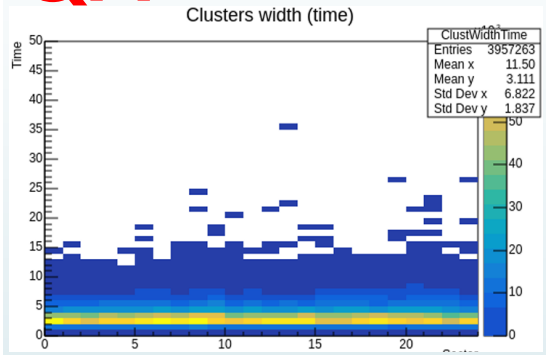
QA data rely on such information as:

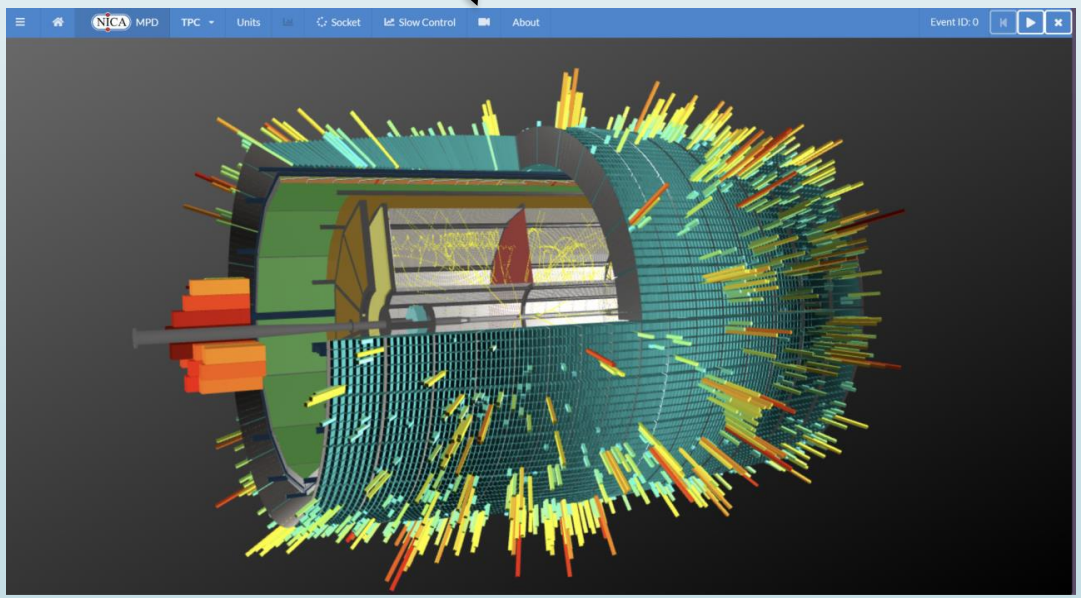
- QA histograms + warnings
- Event Display

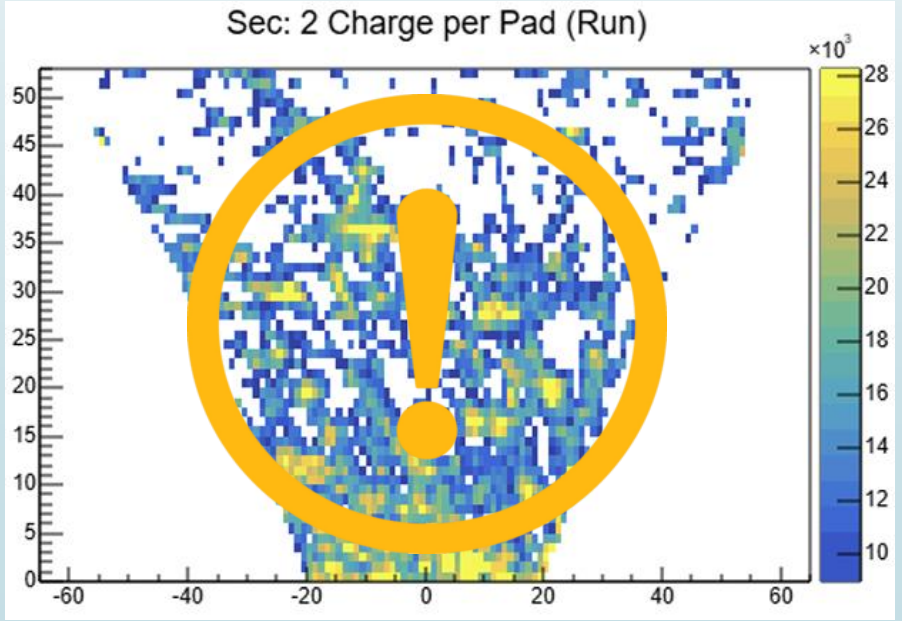


TPC QA



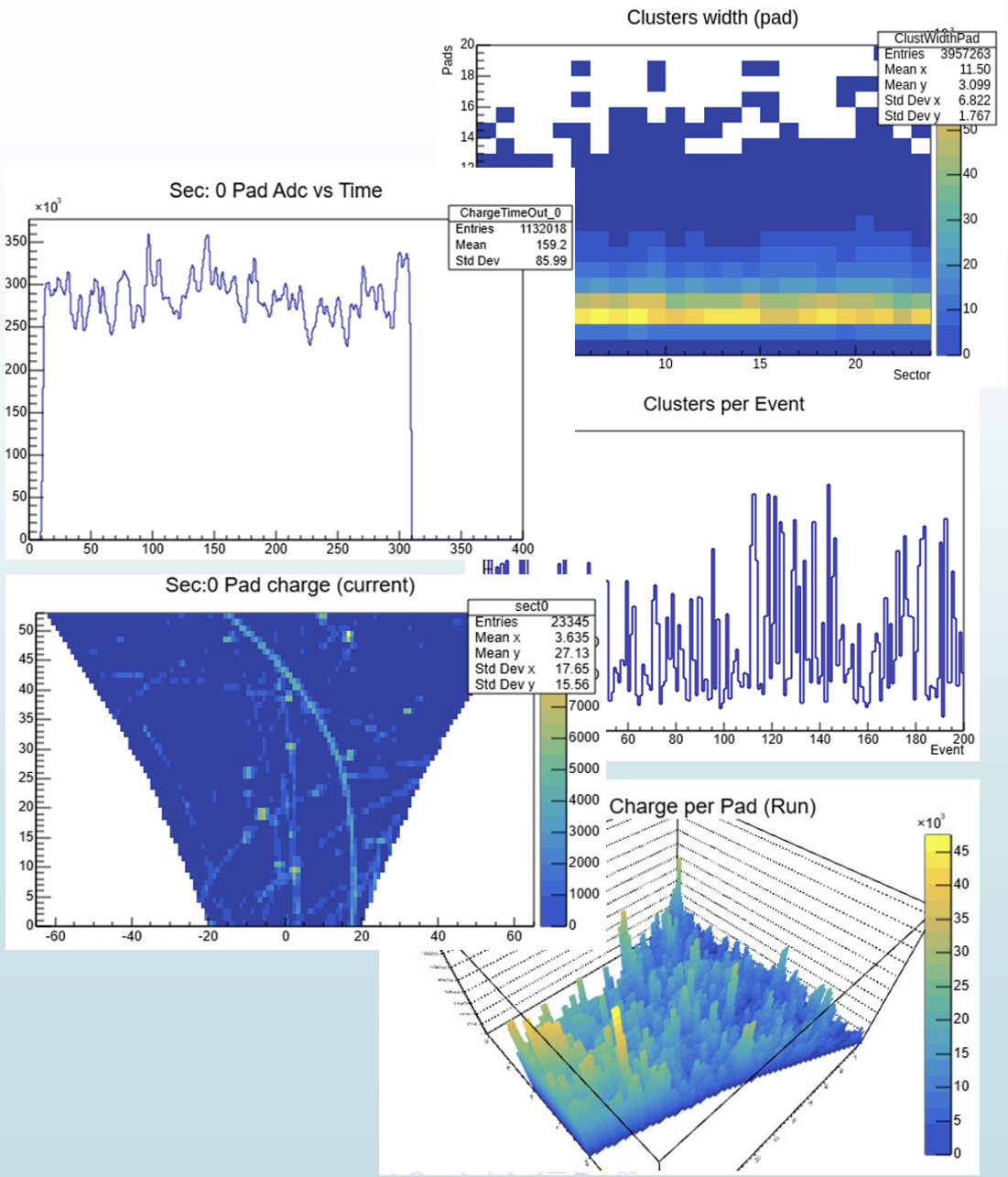




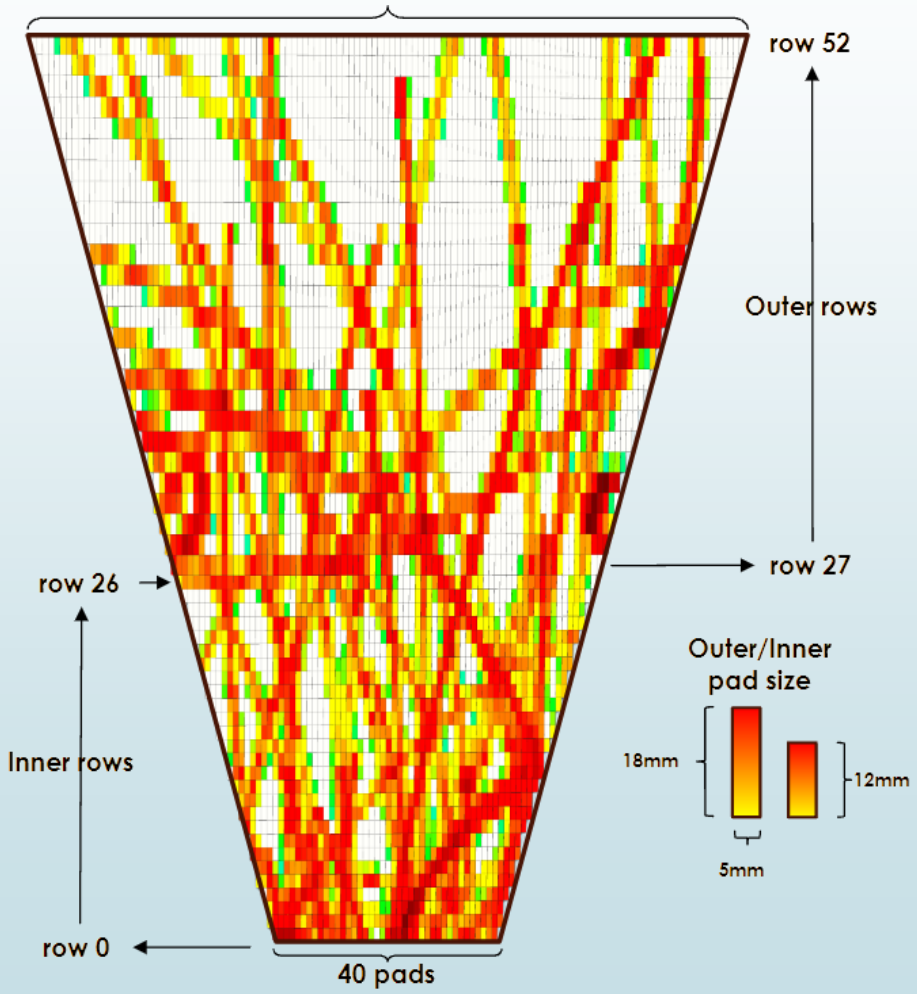


TPC QA histograms

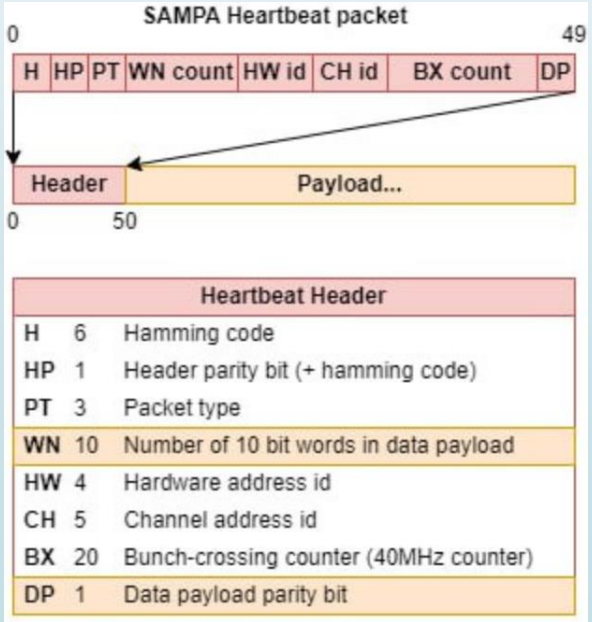
- Inner pads ADC per sector – 24
- Outer pads ADC per sector – 24
- Inner pads ADC per timebucket – 24 (per each sector)
- Outer pads ADC per timebucket – 24 (per each sector)
- Inner pads ADC for current event – 24 (per each sector)
- Outer pads ADC for current event – 24 (per each sector)
- General clusters information – 6
- **Total – over 150 histograms**



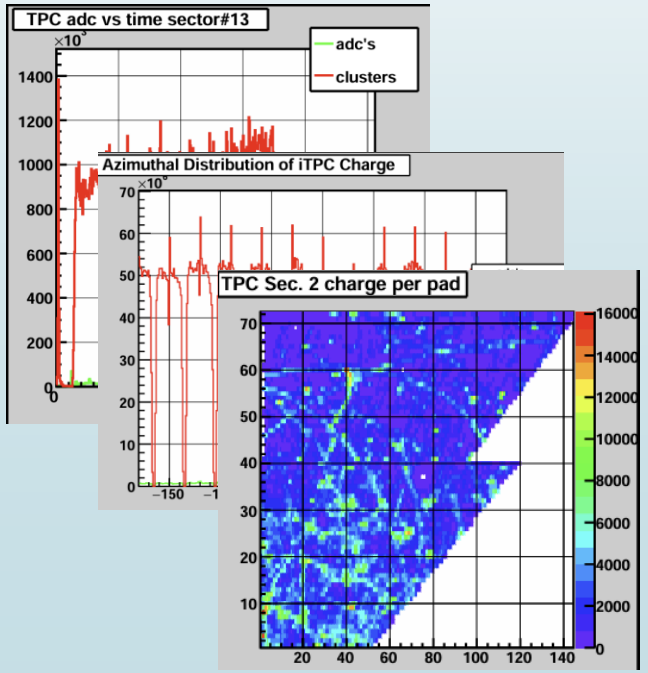
TPC sector data QA



TPC raw data are received in the **SAMPA chip format** from the detector readout. After decoding, this stream is transformed into TPC digits, which represent the ADC signal recorded for each pad sensor and are used as the input for further QA and clustering algorithms.

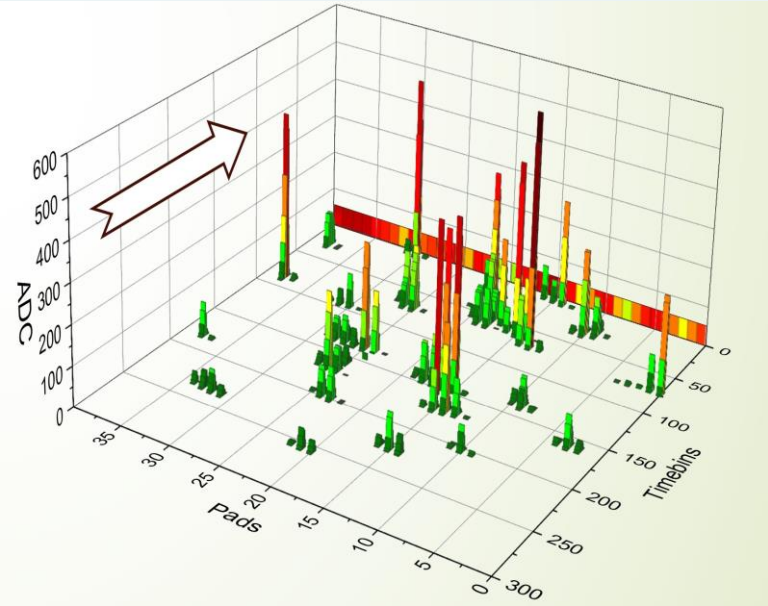


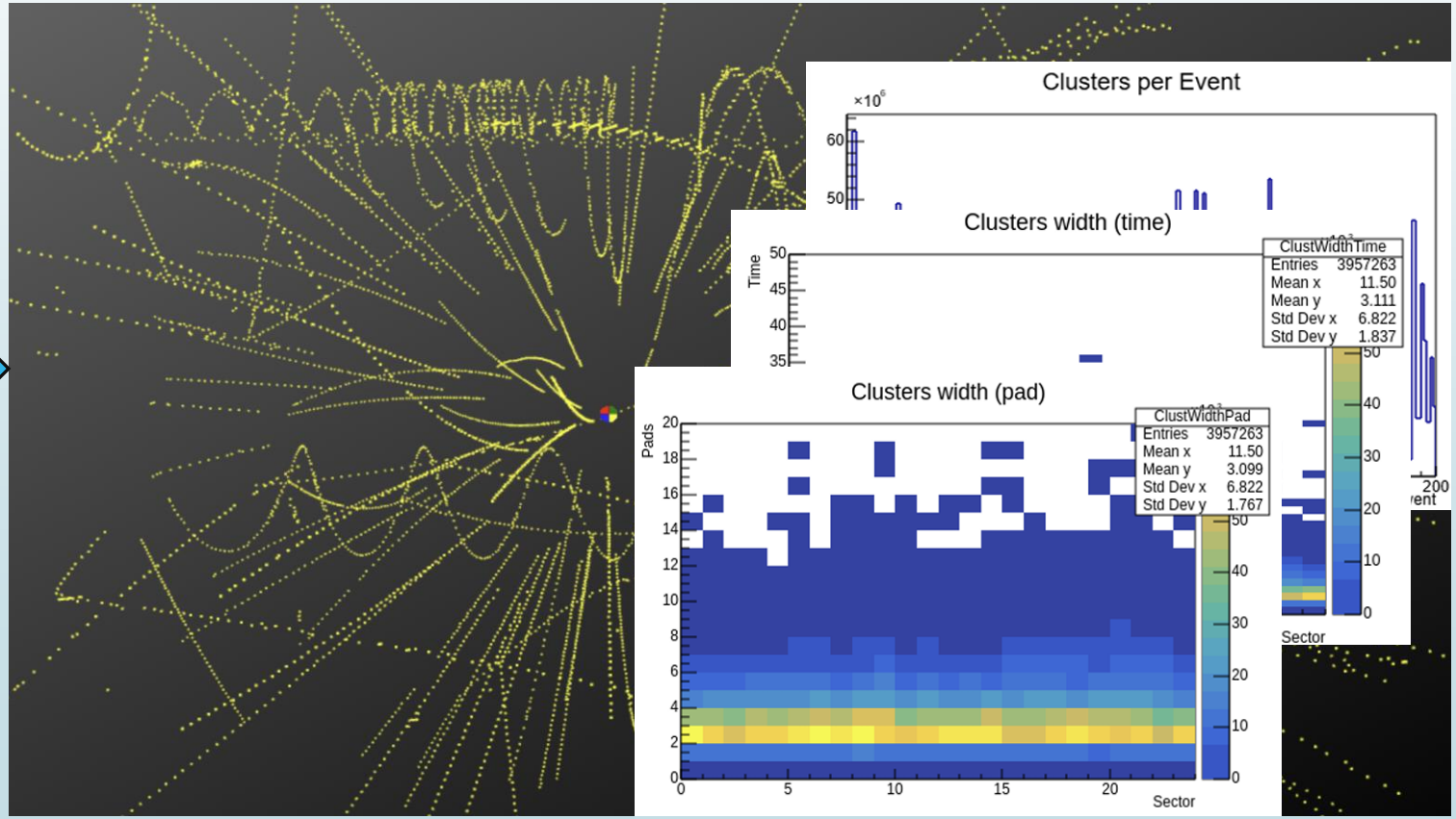
Heartbeat Header		
H	6	Hamming code
HP	1	Header parity bit (+ hamming code)
PT	3	Packet type
WN	10	Number of 10 bit words in data payload
HW	4	Hardware address id
CH	5	Channel address id
BX	20	Bunch-crossing counter (40MHz counter)
DP	1	Data payload parity bit



TPC clustering data QA

The clustering stage converts raw TPC digits into spatial hit candidates that can already reveal the event structure. This step is essential because it reduces raw detector data to a form suitable for both event display and higher-level QA metrics.

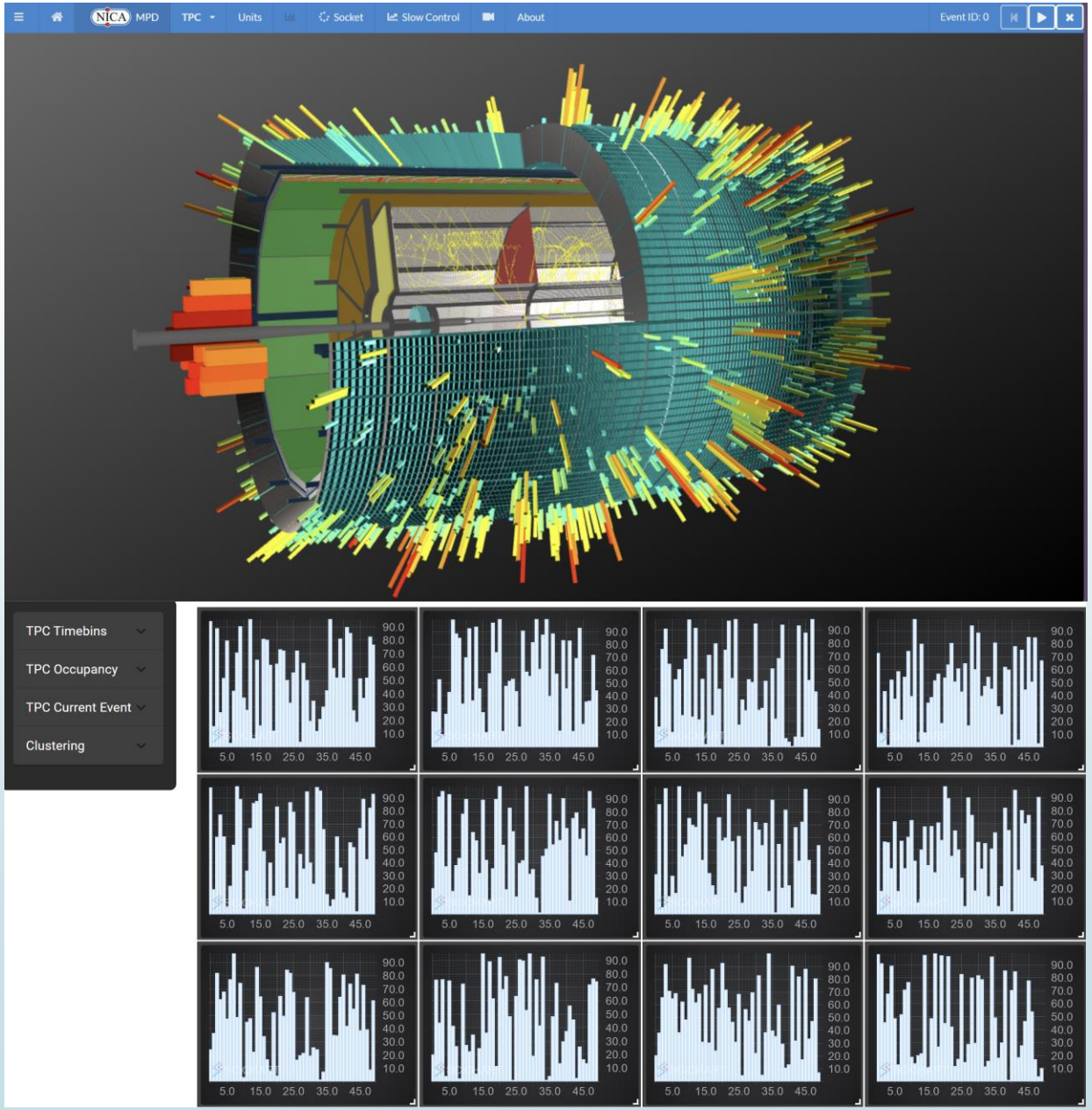


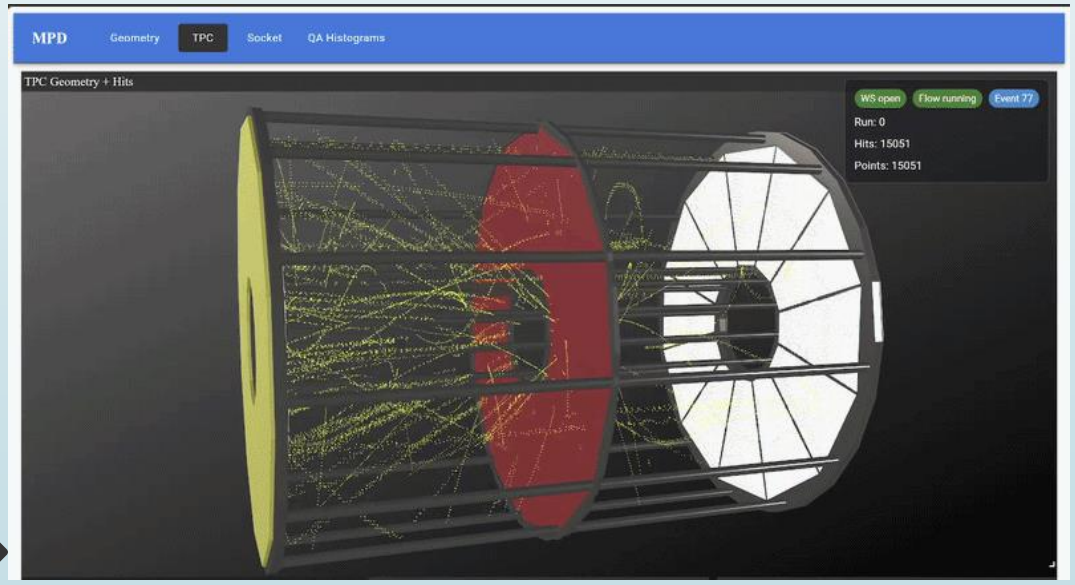


QA + Event Display

Technologies:

- NodeJS - runtime environment
- React - user interface
- WebGL+ThreeJS - 3D graphics



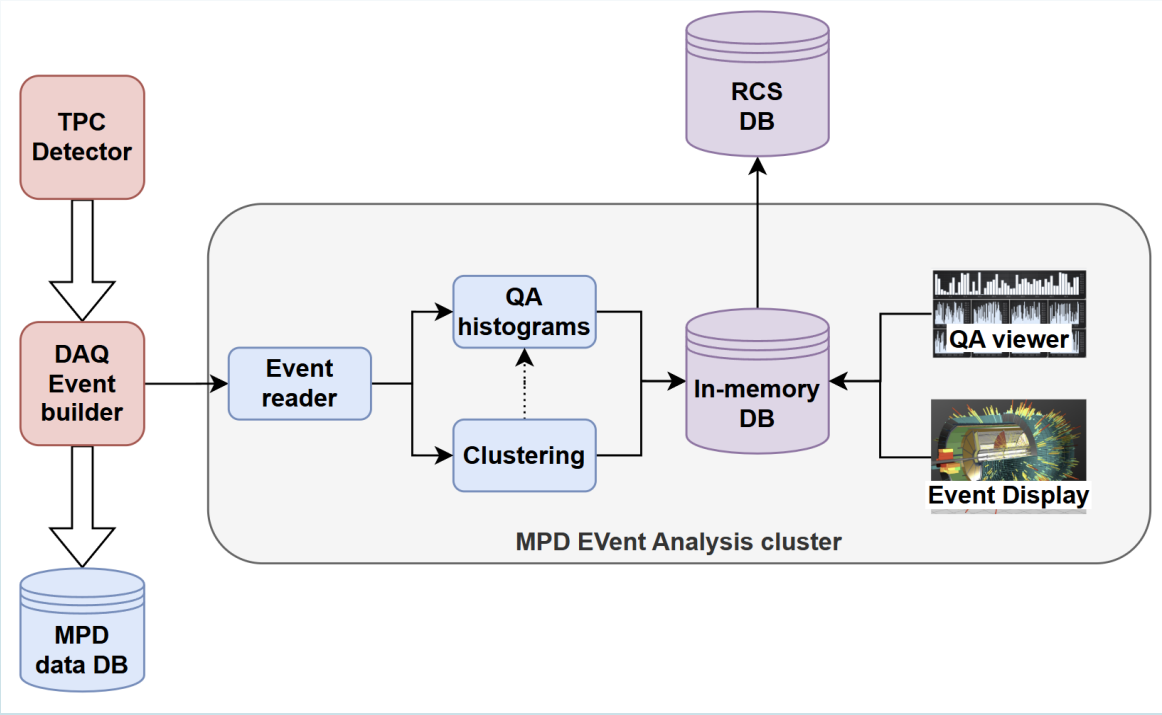


Main data handling flow

The online monitoring pipeline receives raw events from the **TPC detector** through the DAQ event builder.

These events are processed inside the online cluster by dedicated microservices:

- **Clustering** algorithm reconstructs 3D hits
- **QA histograms –** quality assurance
- **In-memory database** – stores information about the last **1000 processed events**.
- **Event Display** enables online data visualisation during experiment
- **QA viewer** – run data verification

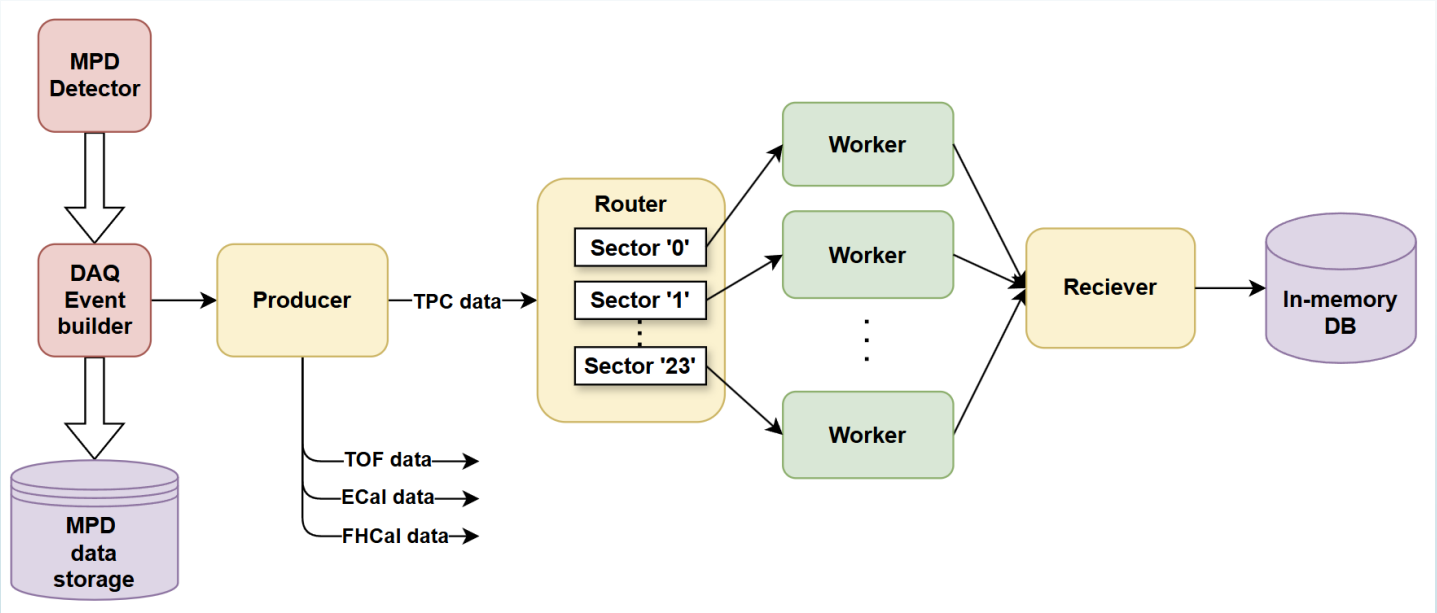


Distributed TPC data processing (**V. Krylov report**)

To keep the processing latency low, the workload is divided into **independent tasks (workers)** that can run simultaneously on different nodes.

The event stream is processed in parallel by many workers, which makes the pipeline scalable but also harder to manage manually.

Therefore, an **orchestrator is required** to control worker lifecycle, scaling, and fault recovery.

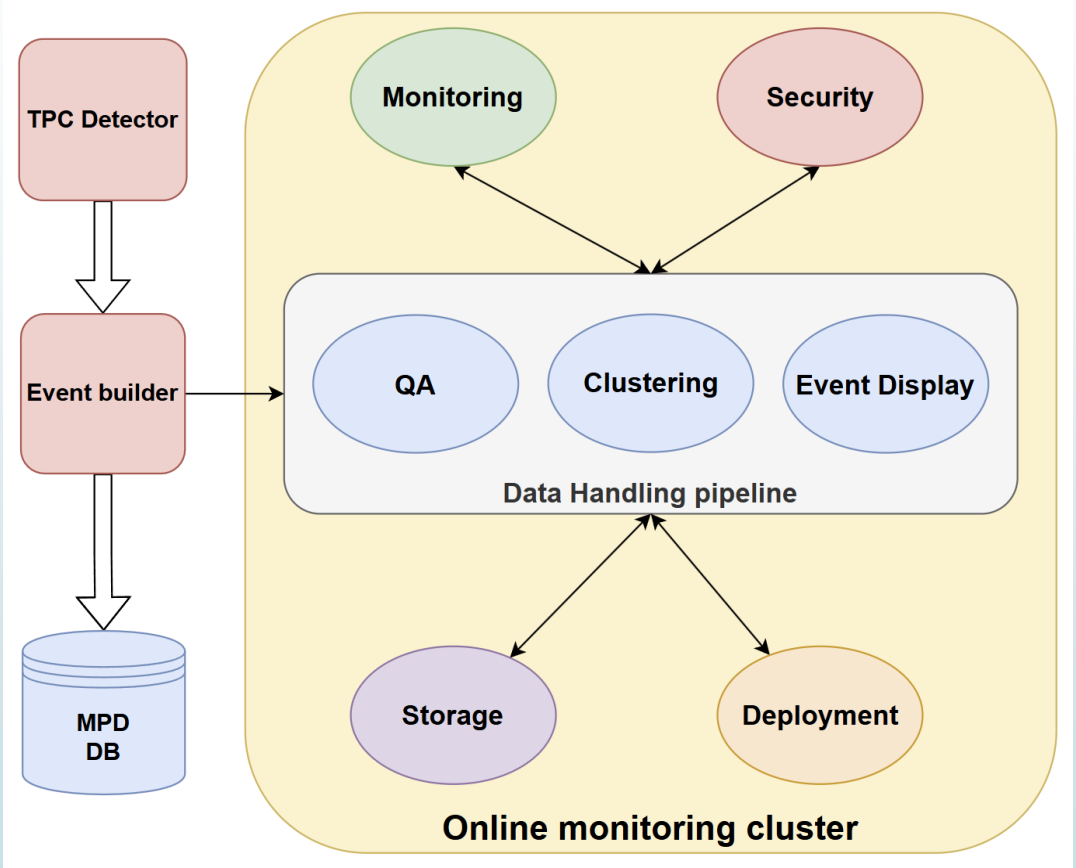


Kubernetes orchestration (**A. Budnikov report**)

In this architecture, Kubernetes provides **cluster-level orchestration** for the data-handling pipeline, including:

- placement of containerized QA, clustering, and event-display workloads
- automatic restart on failure
- controlled rollout of updates.

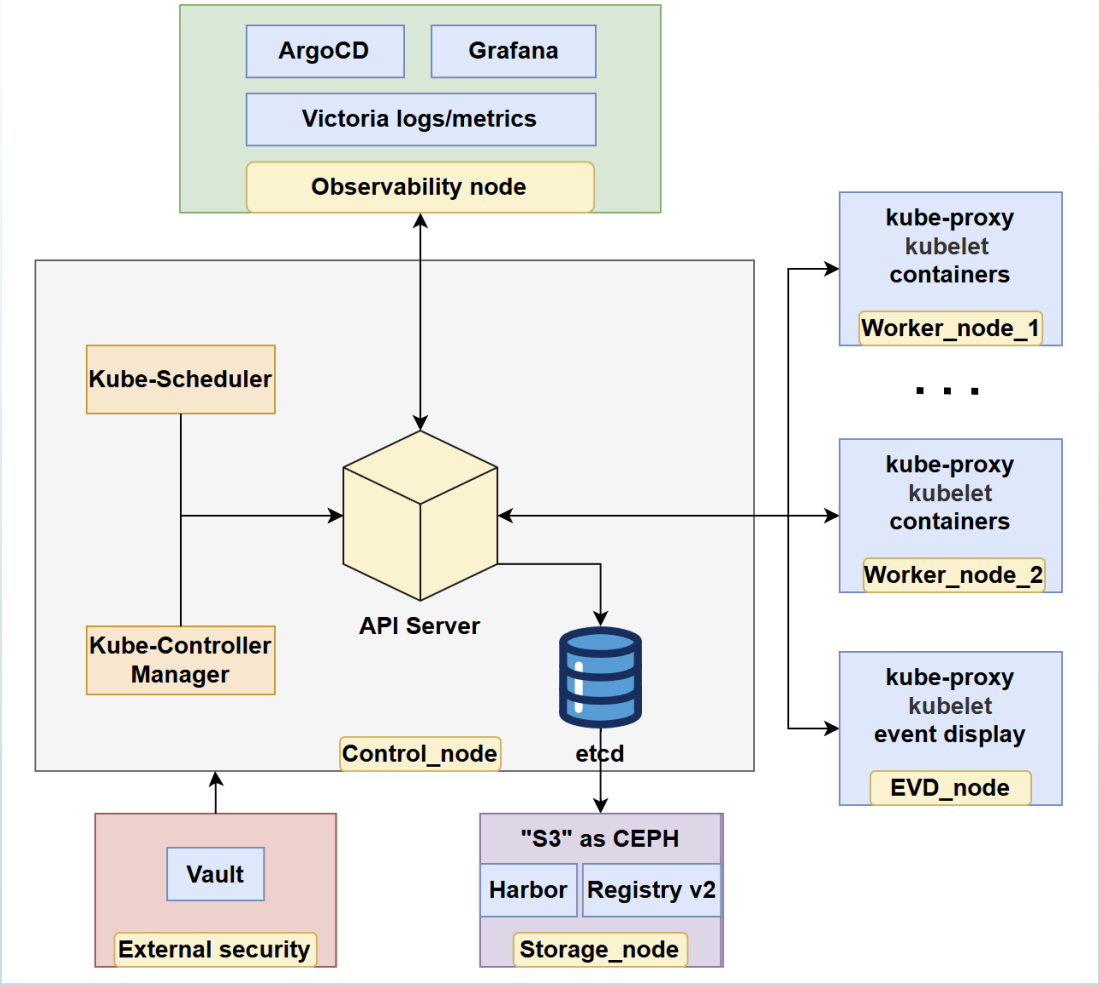
It also unifies auxiliary services such as **observability**, **storage**, **security**, and **deployment** into a single operational platform for online monitoring.



Kubernetes cluster architecture (**A. Budnikov report**)

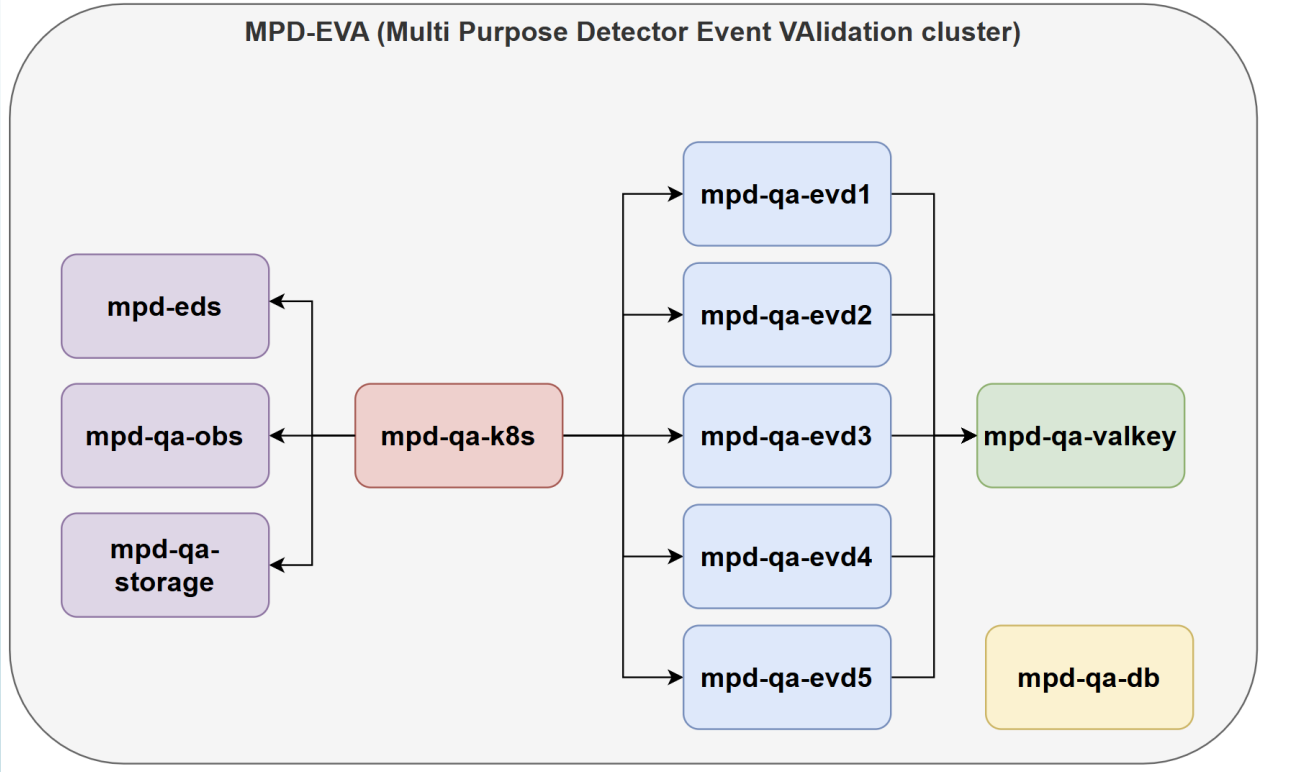
The system is organized as a multi-node Kubernetes cluster for MPD experiment:

- **Control plane** - manages orchestration and cluster state.
- **Worker nodes** - run the processing services.
- **Observability node** - collects metrics and logs.
- **Event display node** - provides 3D representation on detector sensor responses.
- **Storage services (Harbor)** are isolated on a separate node.
- **Vault node** - provides centralized secret management.



Current cluster status

№	Node	Description
1.	mpd-qa-k8s	K8s control plane
2.	mpd-eds	Web Event Display
3.	mpd-qa-evd1	Worker node
4.	mpd-qa-evd2	Worker node
5.	mpd-qa-evd3	Worker node
6.	mpd-qa-evd4	Worker node
7.	mpd-qa-evd5	Worker node
8.	mpd-qa-valkey	In memory fast DB
9.	mpd-qa-obs	Observability stack
10.	mpd-qa-storage	Local images registry
11.	mpd-qa-db	Mpd runDB (Postgress)



Metrics dashboards

- Metrics dashboards provide the operator with a **compact view of pipeline health**, including event rate, processing latency, resource usage, and service availability. They help detect performance degradation early, before it appears as missing events or delayed DQA output.



Logs

- Logs complement metrics by showing **detailed runtime events** such as decoding failures, network errors, crashed services, and restart activity. For online monitoring, they are the main tool for fast root-cause analysis when the pipeline behaves incorrectly.



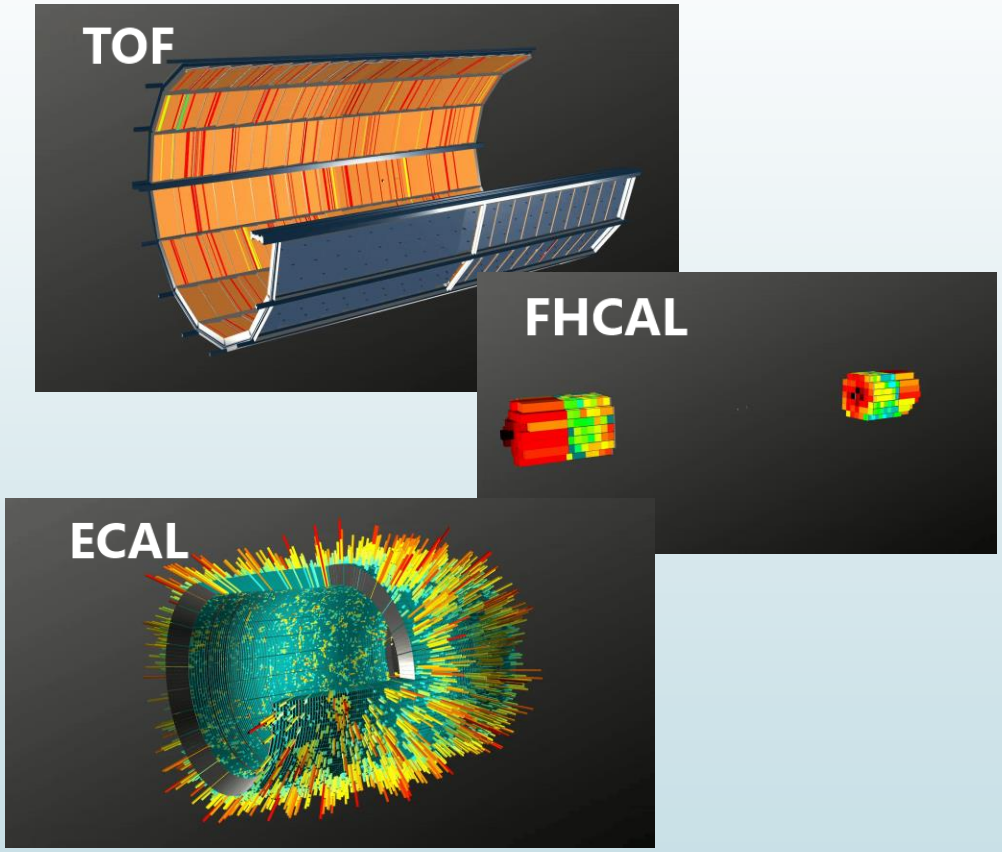
Future plans and current tasks

Current tasks:

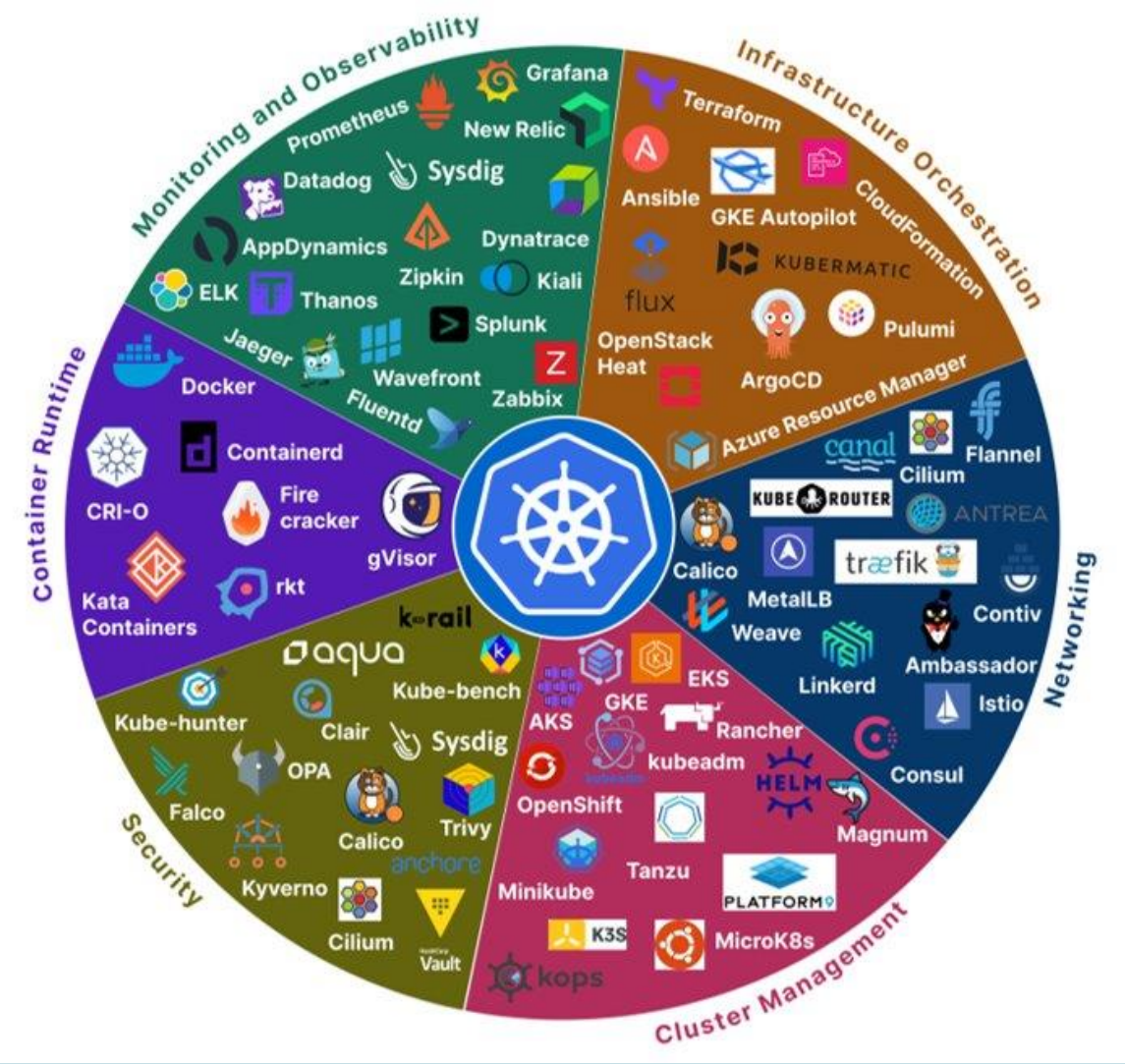
- Test real TPC data events and connection with DAQ cluster
- Discuss Event Display and QA viewer design requirements

Future plans:

- Extend data handling and monitoring to all 1-st stage detectors (TOF, ECal, FHCal, FFD)
- Add security for the project
- Write unit tests for the main data handling pipeline
- **Audit project requirements and limitations (expert)**



Thank you for your attention!



System requirements

The system must reliably run a distributed set of microservices that process data streams with predictable latency and stable throughput.

It needs to scale individual components horizontally, **recover automatically from failures**, and keep the cluster in a consistent, reproducible state during data-taking.

Observability and centralized logging **are required** to maintain operational transparency and protect internal services.

Table 3.1: Estimation of the necessary disk space and power of PCs.

Parameter	Value	Unit
Data rate from L2	4.7	GB/s
Experimental run duration	120	day
Efficiency	60	%
Event rate	6	kHz
Past-future protection	50	%
Total number of events	**$1.87 \cdot 10^{10}$**	event
Mean number of tracks per event	500	track
Mean number of hits per particle (TPC + eCAL)	20	hit
Mean number of bytes per hit	45	byte
Total size of RAW data	**8.4**	PB
Total necessary disk space	**10**	PB
Mean reconstruction time per event by 1K SI2K PC	2	s
Total time per year for 1 PC	7000	hour
Number of 1K SI2K PCs for reconstruction	**1480**	
Total number of necessary PCs	**1800**	

MPD CDR 2006

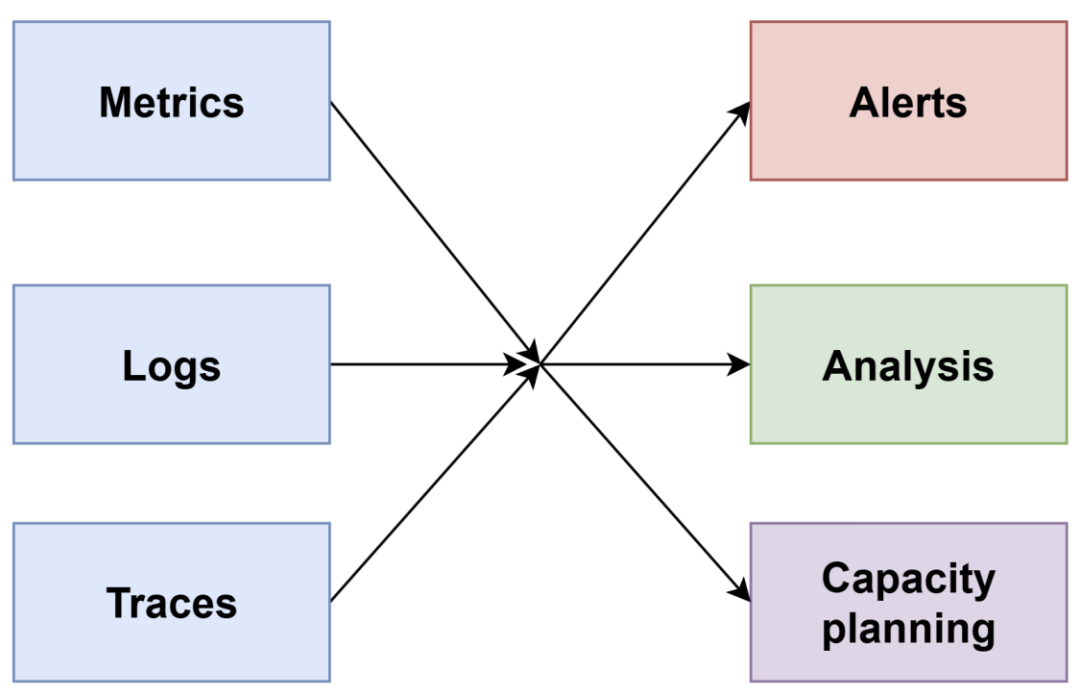
Estimated Run parameters (MPD CDR)

Observability stack

In a distributed pipeline, failures are rarely isolated. A slowdown in workers may appear as queue growth, increased resource usage, and delayed output.

Monitoring lets us correlate these signals. It improves **incident response**, supports **root-cause analysis**, and helps us **scale the system** based on actual workload.

A well-designed observability stack turns raw signals into actionable insight, ensuring that the online cluster remains stable, predictable, and responsive under real experimental conditions.



Requirements — questions the implementation must answer

What end-to-end latency is acceptable from event builder to the shifter screen?

What fraction of the stream is processed continuously?

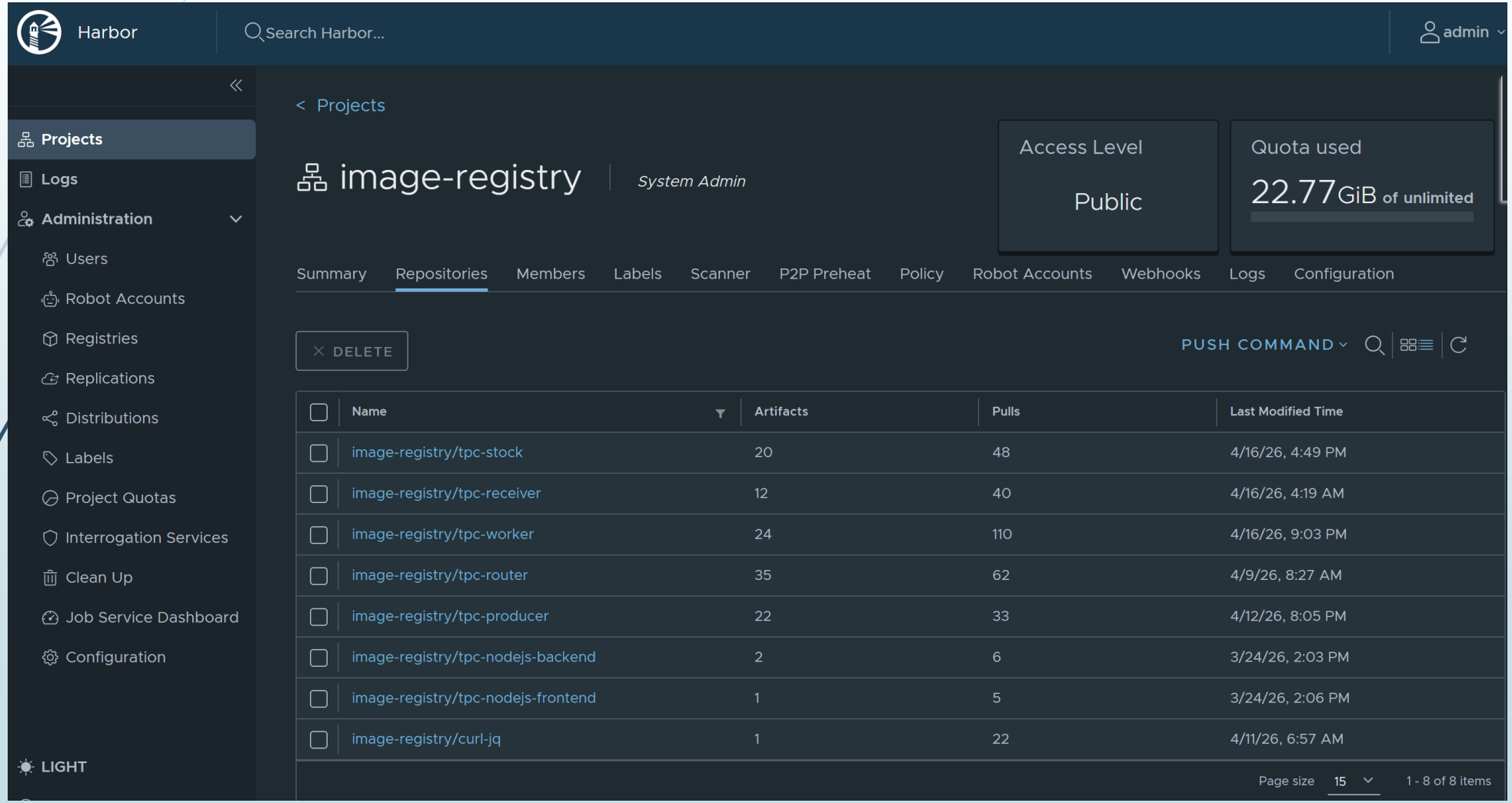
How large should the rolling Valkey window be for operators and experts?

What DQA data format should be stored in long-term storage?

Which metrics, logs and alerts are mandatory before the system is considered “operational”?

Who will be responsible for control room requirements document?

Harbor image registry



Conclusion

- The proposed architecture makes online detector monitoring for the MPD experiment practical and dependable.
- By processing events with a **distributed microservice** design and running services on **Kubernetes**, the system can grow with demand and recover from failures without losing visibility.
- A unified observability stack (metrics, logs, traces) gives operators clear, real-time insight into processing pipelines health.
- We need to **work closely with the DAQ team** to establish a reliable connection to their data flow. Stable access to the event stream is essential for integrating our online monitoring pipeline and ensuring that DQA receives the information it needs in real time.