

BM@N Offline Visualization Software and Other Services

Igor Dunaev

the MIPT team for the BM@N collaboration

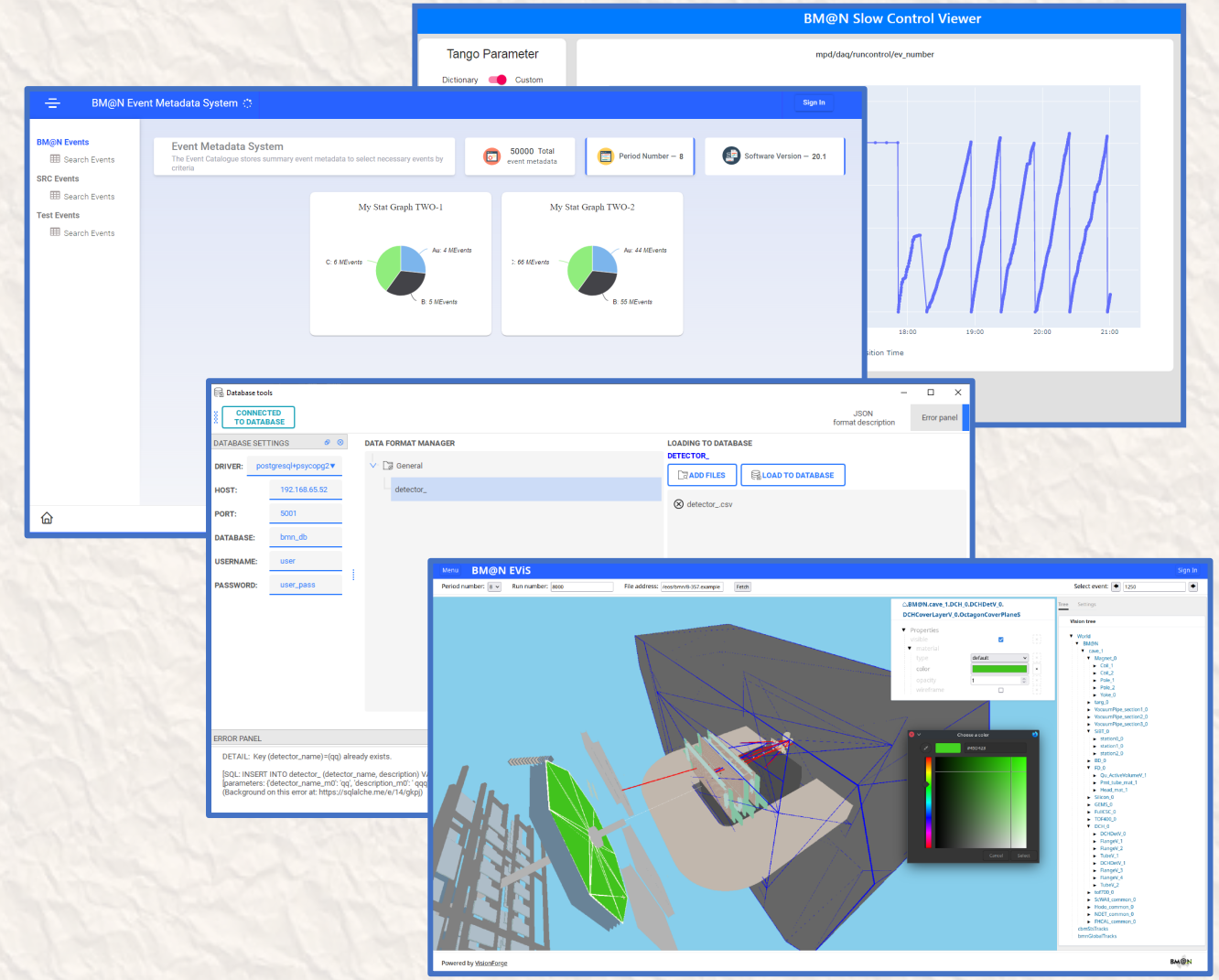


MIPT Software for BM@N – Team

Supervision: T. A.-Kh. Aushev

Team members:

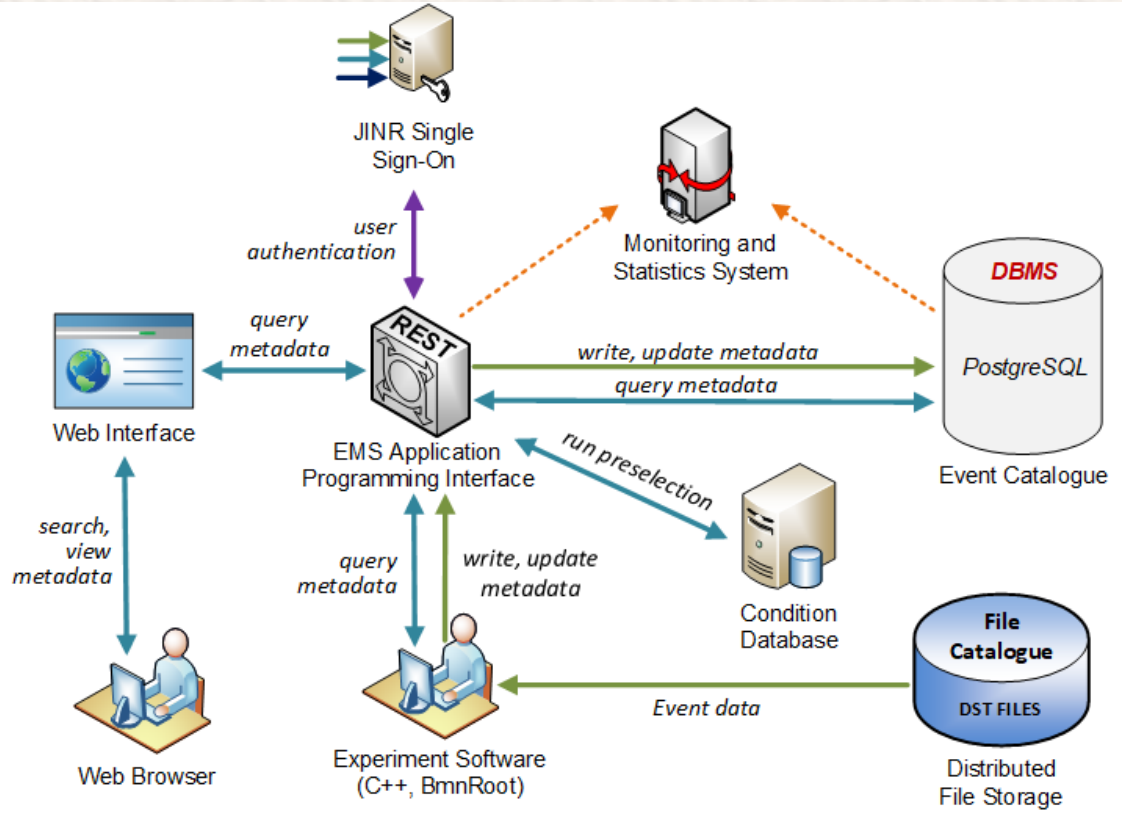
- P. Klimai
- A. Nozik
- I. Dunaev (PhD st. 1y) - Event visualization system
- O. Nemova (PhD st. 1y) - Monitoring system
- A. Degtyarev (PhD st. 3y) - Event metadata system



Current Projects Summary

Project	URLs
Event Metadata System	https://git.jinr.ru/nica_db/emd https://bmn-event.jinr.int/
Next-generation Event Display	https://git.jinr.ru/nica-visualization/visionforge https://git.jinr.ru/nica-visualization/visapi https://git.jinr.ru/nica-visualization/evis
Monitoring Service	https://git.jinr.ru/nica-computing/mon-service-deploy https://mon-service.jinr.ru
REST API and Web interface for slow control system	https://git.jinr.ru/nica_db/tango_web https://git.jinr.ru/nica_db/tango-rest

BM@N Event Metadata System (1)



- Event Metadata System
  - Event Catalogue is based on PostgreSQL
  - Integrates with BM@N Condition database
  - REST API and Web UI developed based on Kotlin multiplatform
  - Configurable to support different metadata
  - ROOT macro to fill in the catalogue
  - Automatic deployment
  - High Availability solution available
  - Statistics collection and display
  - Monitoring

For more details:

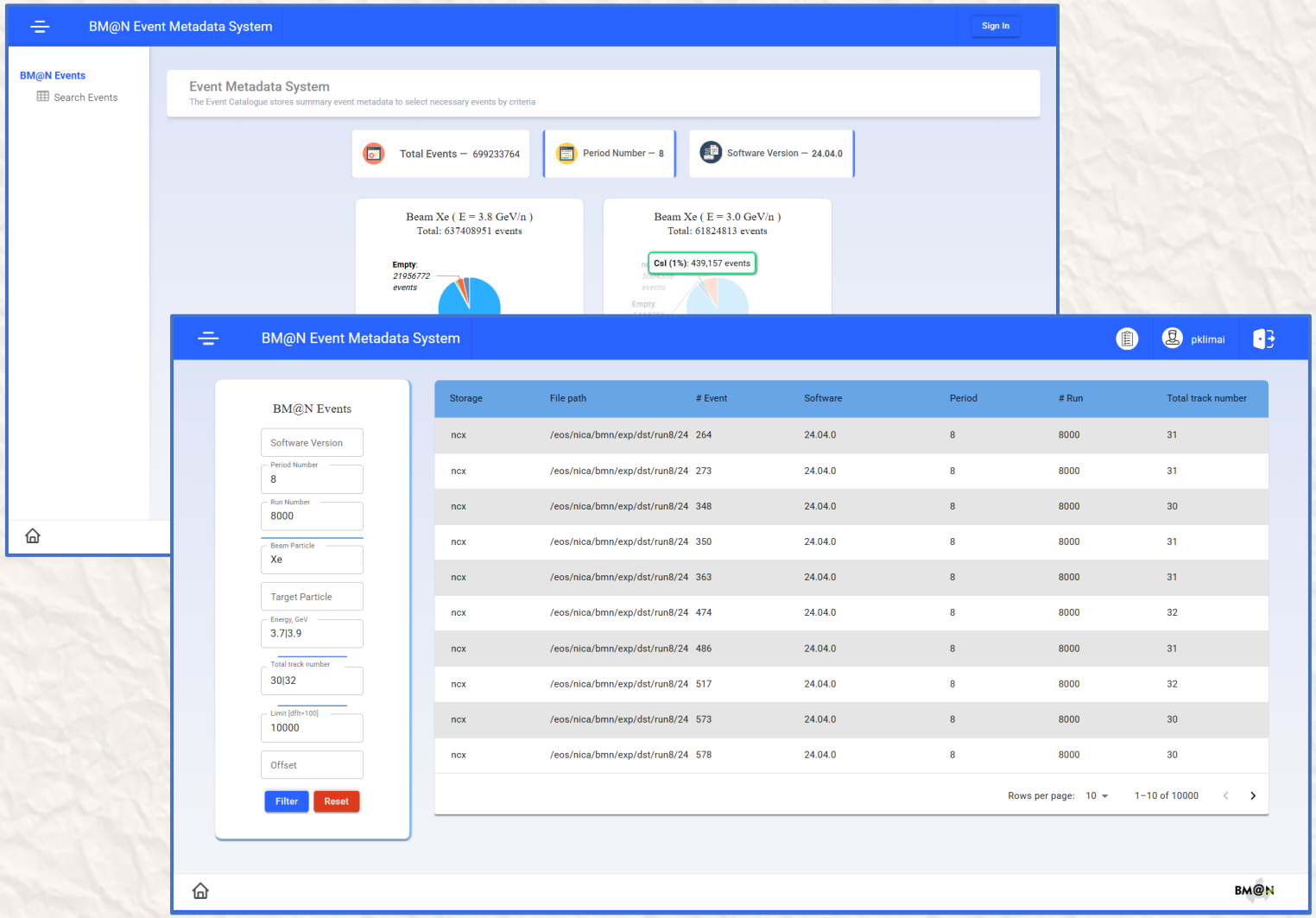
E. Alexandrov, I. Alexandrov, A. Chebotov, A. Degtyarev, I. Filozova, K. Gertsenberger, P. Klimai and A. Yakovlev, "Implementation of the Event Metadata System for physics analysis in the NICA experiments", J. Phys.: Conf. Ser. 2438, 012046 (2023).

Recent paper:

K. Gertsenberger, P. Klimai, I. Dunaev, A. Degtyarev, O. Nemova, "Improving Availability and Performance of the Event Metadata System for the BM@N Experiment, Phys.Part.Nucl.Lett. 22, 1022-1025 (2025).

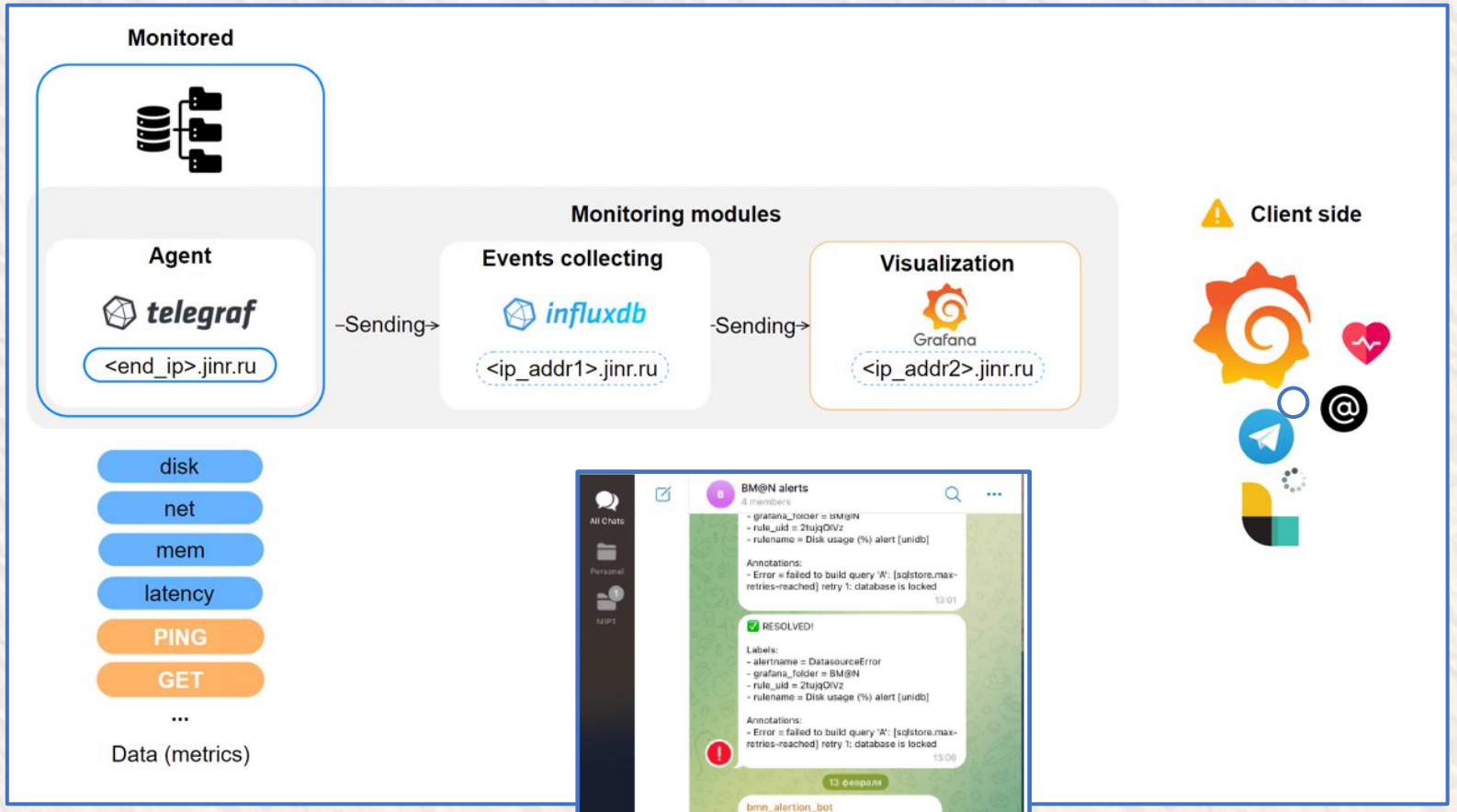
BM@N Event Metadata System (2)

- Latest version deployed at https://bmn-event.jinr.int/ with about 700M records of Run 8 currently in it
- Performance improvements for bulk writing of new events
- Improved error processing and display
- Statistics collection script is now part of main repository
- Bug fixes per request
- Some extra services for EMS in development



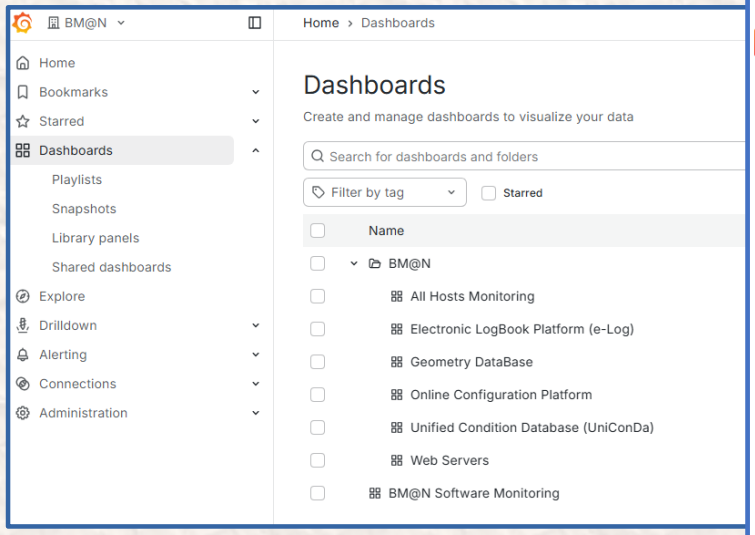
BM@N Monitoring System (1)

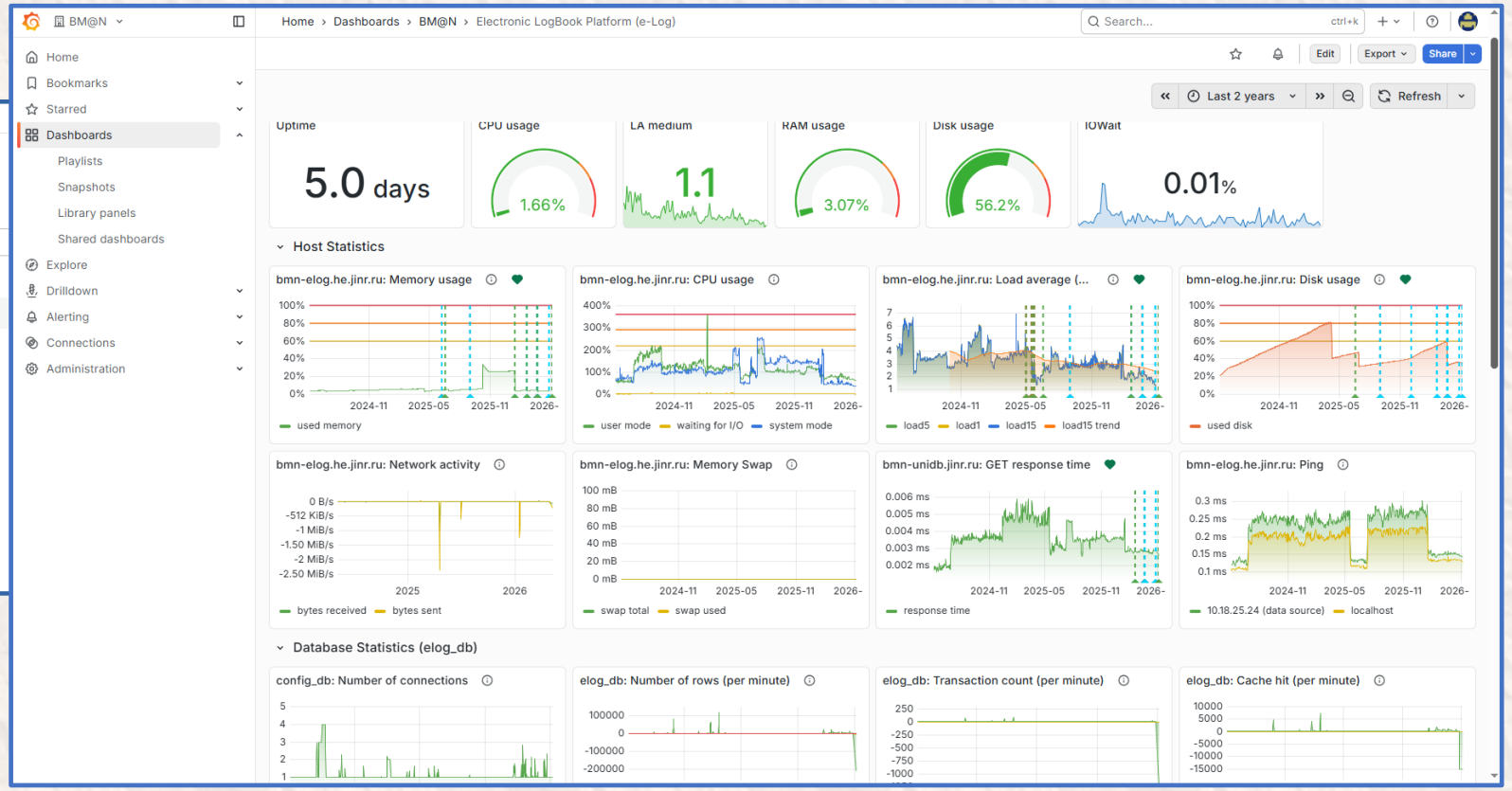
- Monitoring system and status
  - Using TIG (Telegraf + InfluxDB + Grafana) stack
  - Automated deployment of components with Ansible
  - Failure alerting with Grafana / Telegram



BM@N Monitoring System (2)

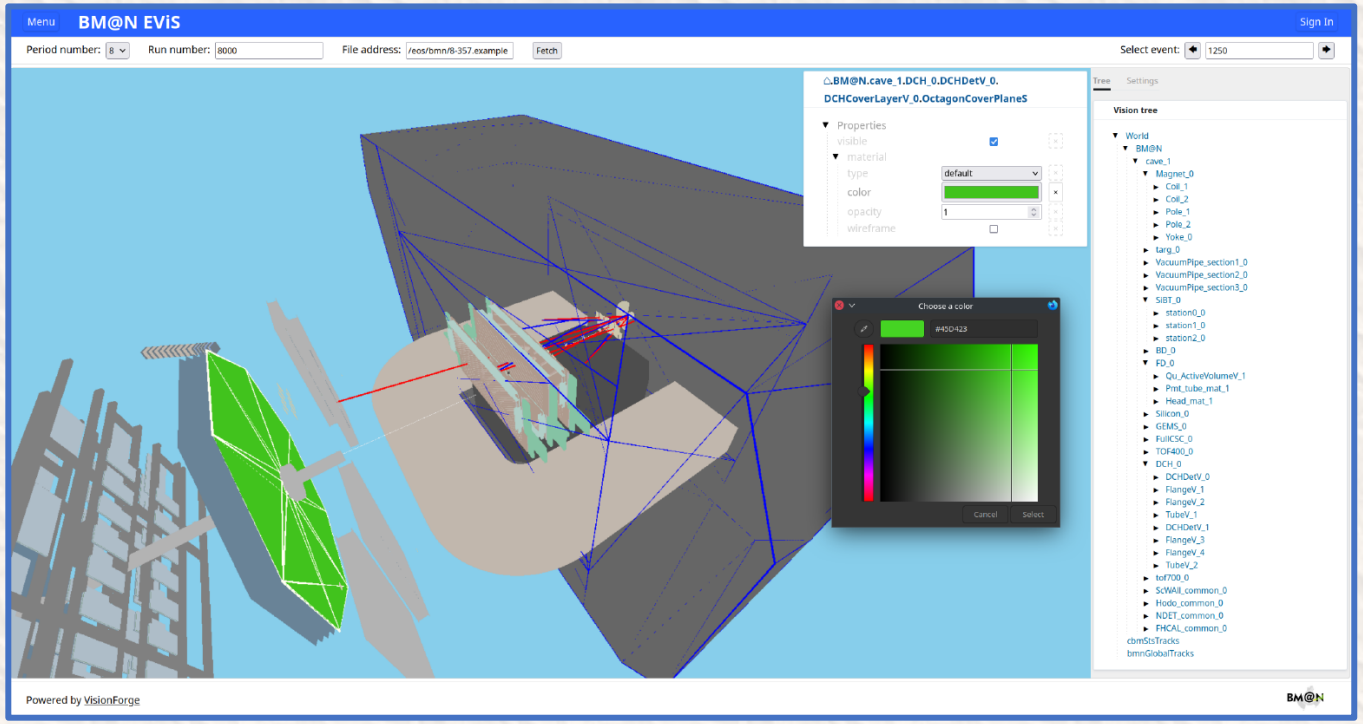
- The system GUI is available at: https://mon-service.jinr.ru
- Dashboards created for multiple monitored nodes





Visualization System

- Kotlin application (JVM) + browser user interface;
- Local filesystem browsing (simulation and DST files);
- Loading events from a remote source;
- Support for online geometry loading.





VisionForge overview

- VisionForge – platform for creating next-gen visualization systems
- Distributed dynamic system
  - Visualization model can be created on one node, transferred to another node and rendered there
  - Nodes can exchange **updates** to the model
  - Changing one element or attribute only requires sending this small change
- Performance and optimizations
  - BM@N geometry model includes more than 400 000 elements
  - Geometry can be defined as **prototype** that is used by a set of objects, in this case rendering is simplified – only required properties can be changed if needed



VisionForge overview

- See also:
- VisionForge project
  - sciprog.center/projects/visionforge
- VisionForge source code
  - git.sciprog.center/kscience/visionforge/src/branch/dev/
- Alexander Nozik — Unbearable lightness of data visualization in Kotlin full stack
  - https://www.youtube.com/watch?v=uT5j-xOXC3E&ab_channel=JPoint%2CJoker%D0%B8JUGru



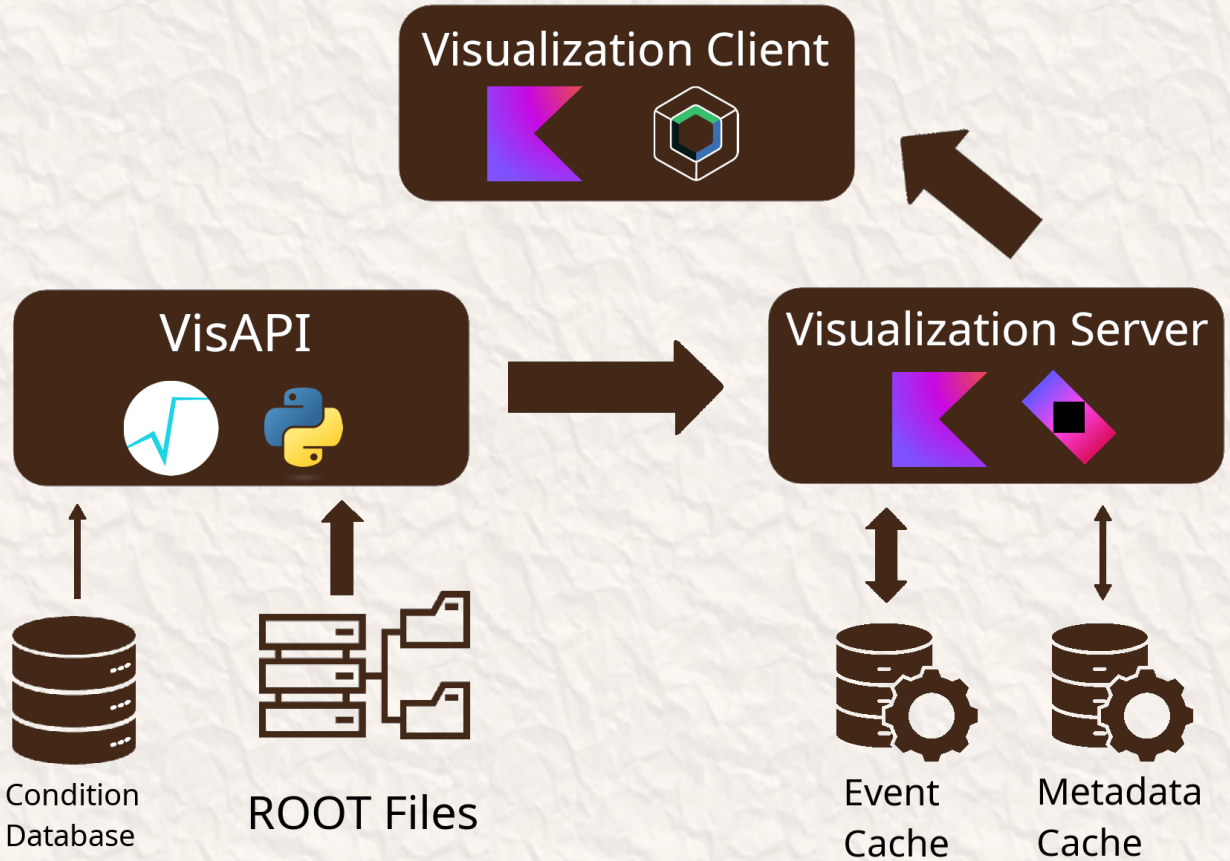
New in VisionForge 0.5.1

- Plotly-kt (Plotly wrappers for Kotlin Compose Multiplatform and Compose-HTML) are included into VisionForge;

- Mostly internal changes:

- Simplification of Vision class logic;
- Unused classes (VisionChildren and VisionProperties) are removed;
- Kotlin vesrion is updated to 2.3;

Previous work



Architecture Updates

Architecture was redeveloped to be more flexible:

- VisAPI (Python server) used as a remote data source replaced with a Kotlin module:
  - VisAPI module is included in the main server by default
  - VisAPI can be deployed on a separate machine if exposing a node with filesystem access is undesirable
- Conversion macros and functions between .root files and Vision classes of VisionForge are implemented:
  - Visions represent only necessary information (coordinates, shapes) with metadata (energy, particle type) required for visualization, being more efficient to store rather that full ROOT JSON strings;
  - Hits, tracks and calorimeter towers are extracted to intermediate data objects. Only tracks and hits are visualized as for now;

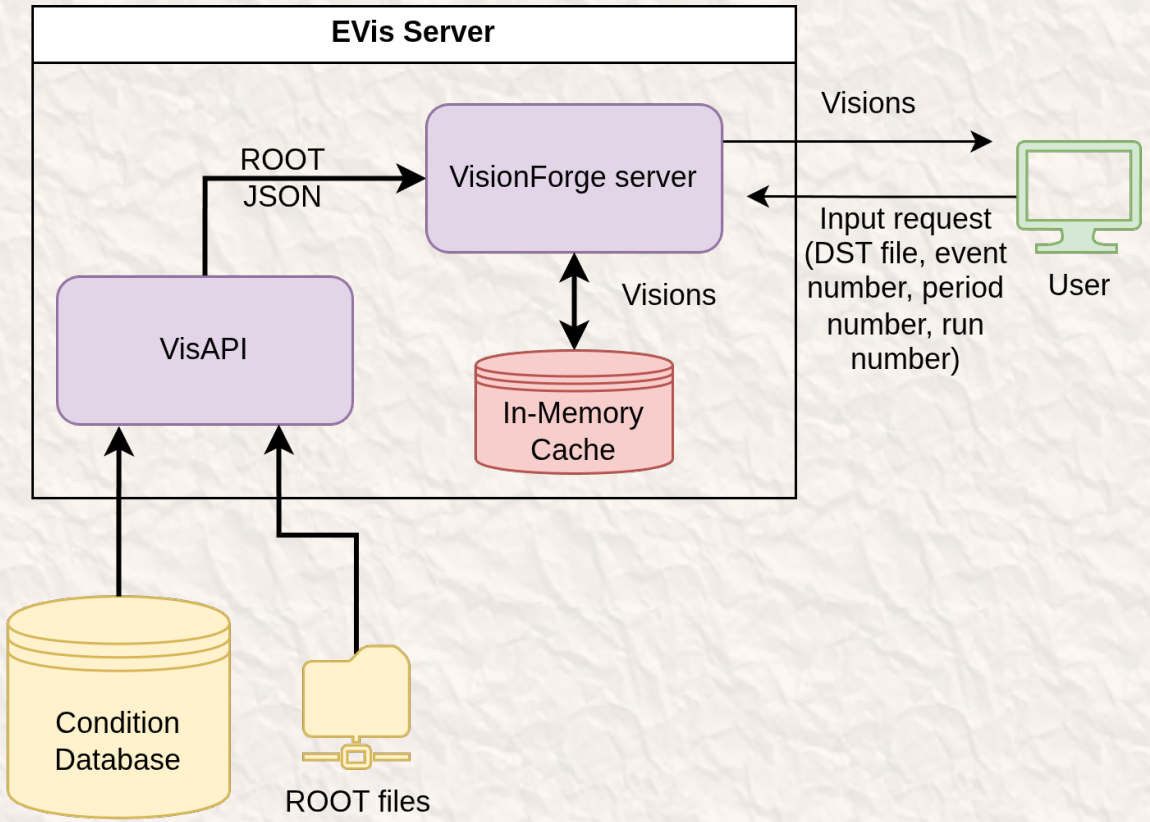


Desktop event display

Local (desktop) event display capabilities added:

- Supports reading from ROOT files with simulated data and reconstructed events;
  - Both geometry and event data are loaded from the files;
- In-Memory performance cache extended with a filesystem-based implementation;
  - Can store events as JSON, gzipped JSON or gzipped CBOR files;
  - Internally events are represented as serialized Vision objects from VisionForge
- This allows for cache persistence for a local application;
- Persistent cache allows using the application for viewing files converted in advance without ROOT installed on the host computer.
- Combined with cross platform capabilities of the main application, it allows running event display on Windows machines.

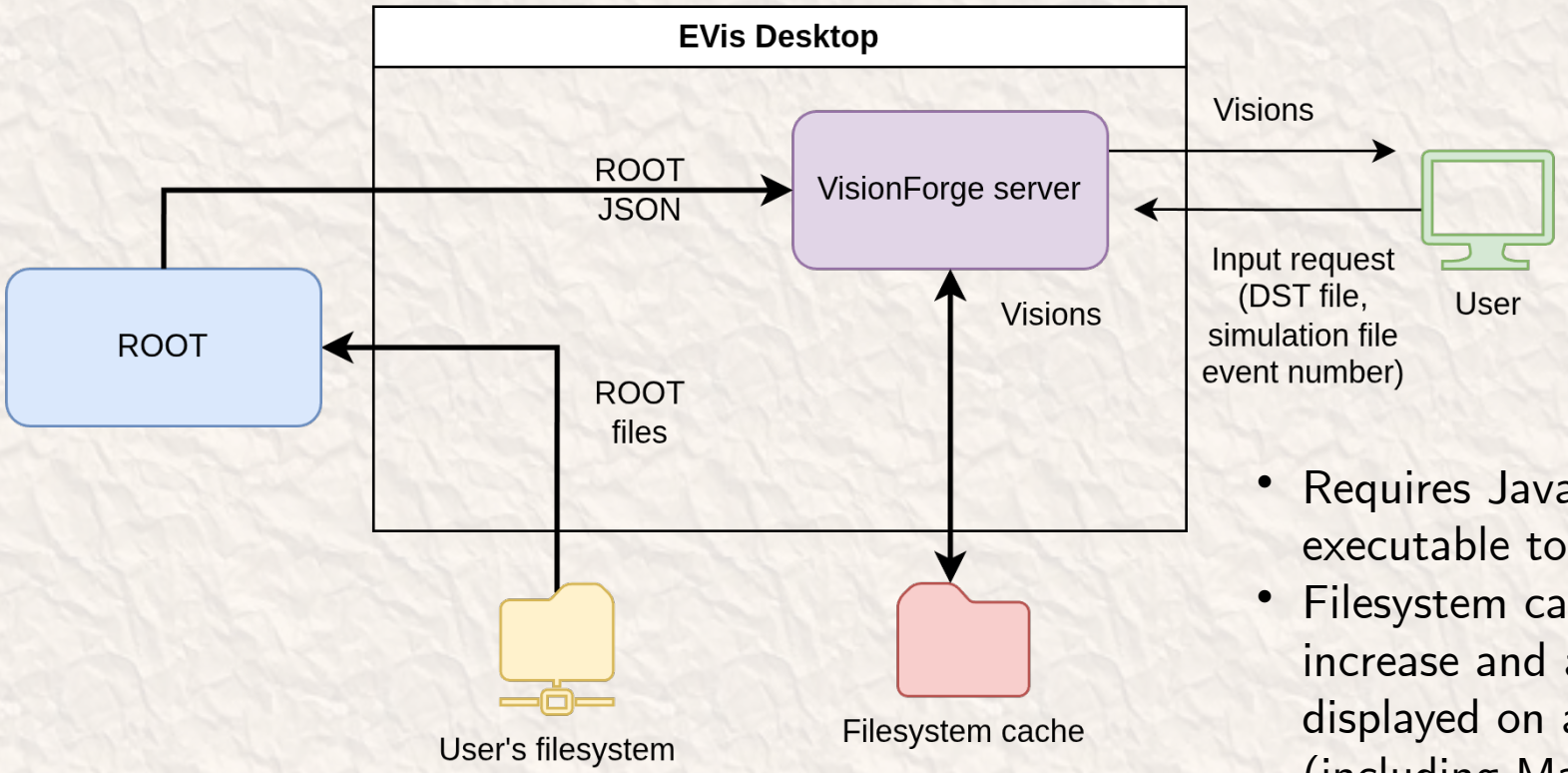
Web Application



- JVM server; UI is a Compose web application hosted on the server;
- Load events and geometry from a remote storage (server filesystem / EOS);
- Uses ROOT executable to convert data from ROOT files to JSON;
- In-memory cache allows for increased performance when browsing continuous event range.

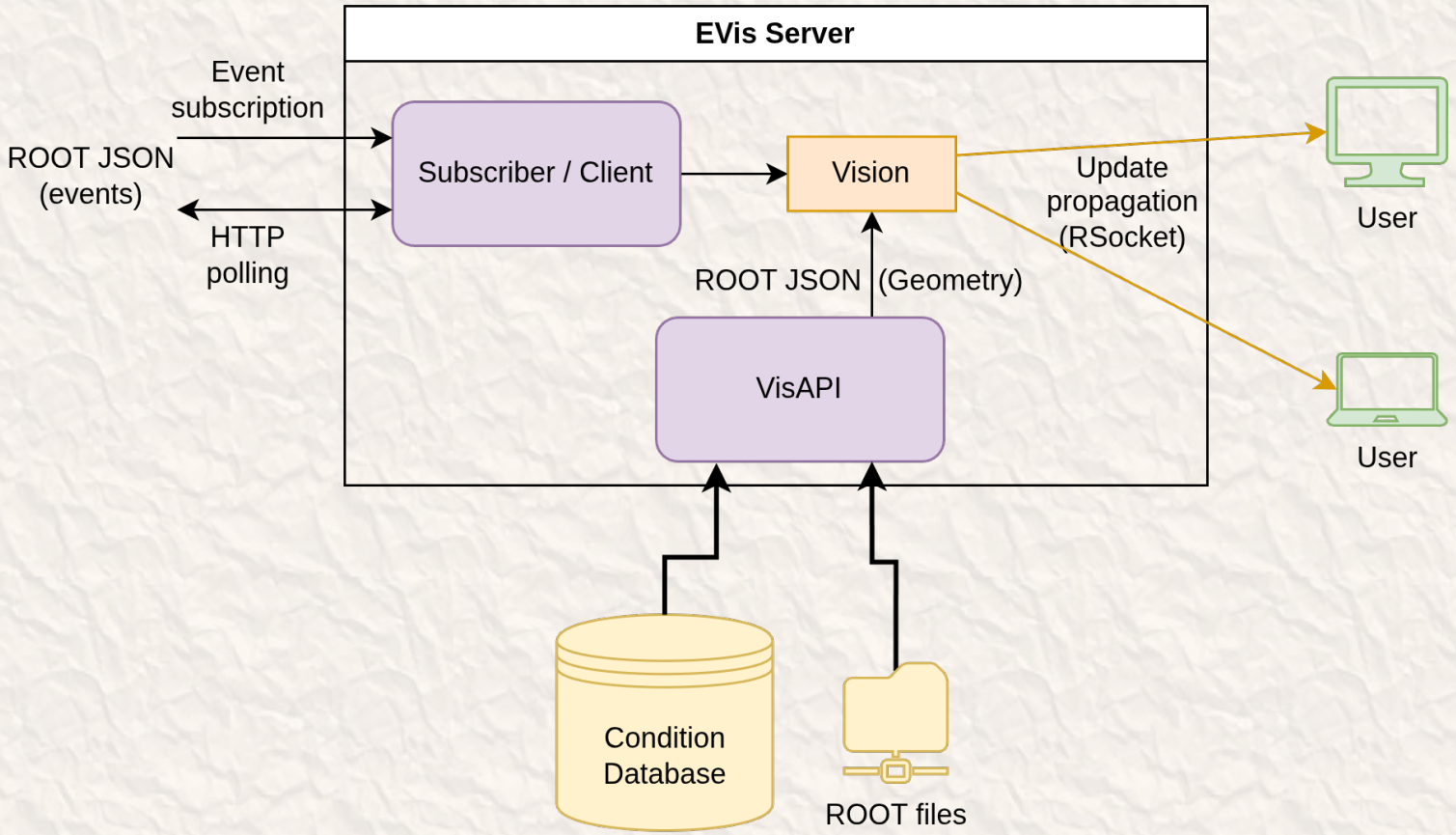
Desktop application

Application with an internal server and same browser UI as the web version.



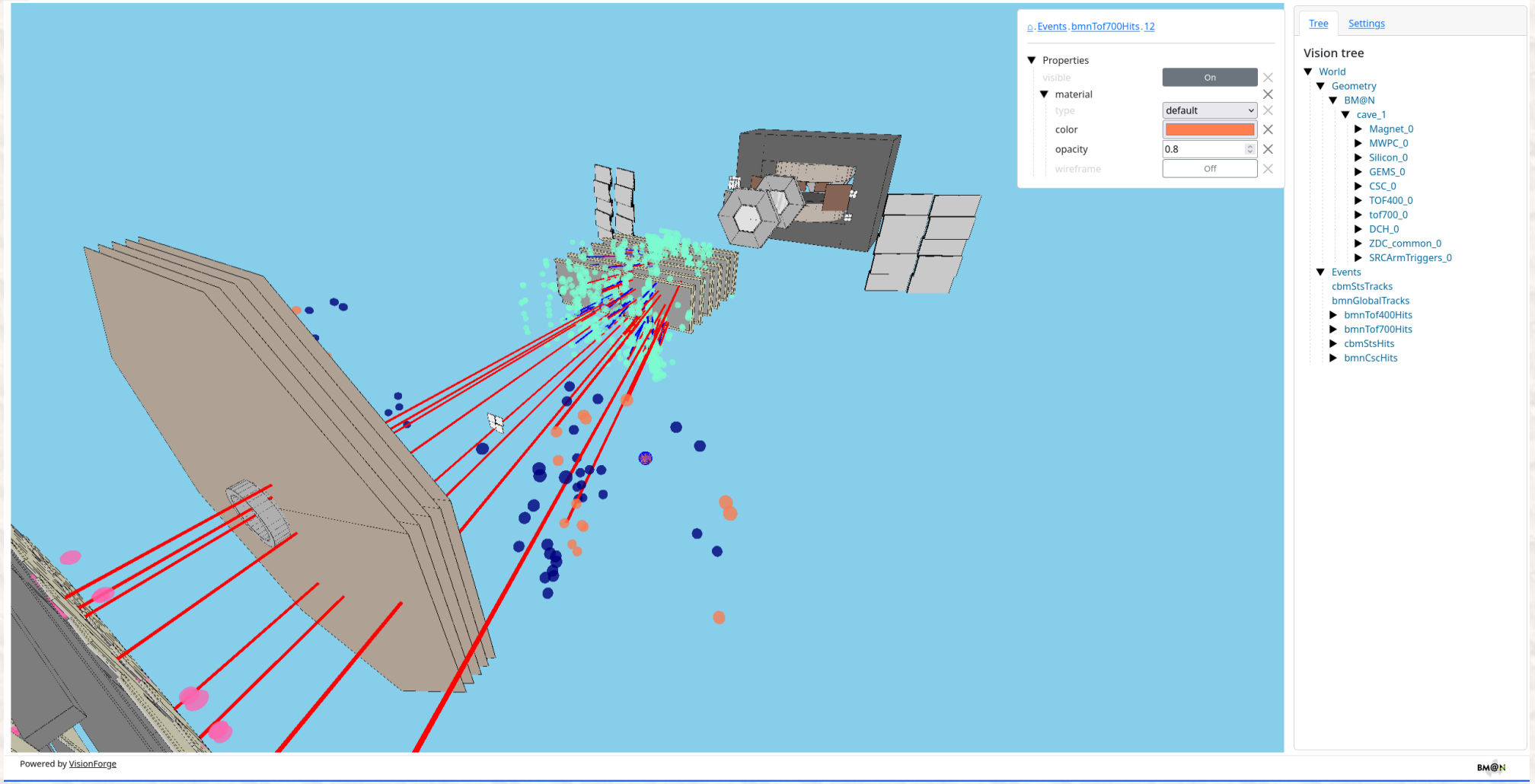
- Requires Java runtime to run and ROOT executable to to data conversion
- Filesystem cache allows for both performance increase and ability to convert data to be displayed on a machine without ROOT (including Mac / Windows).

Planned Monitoring Application

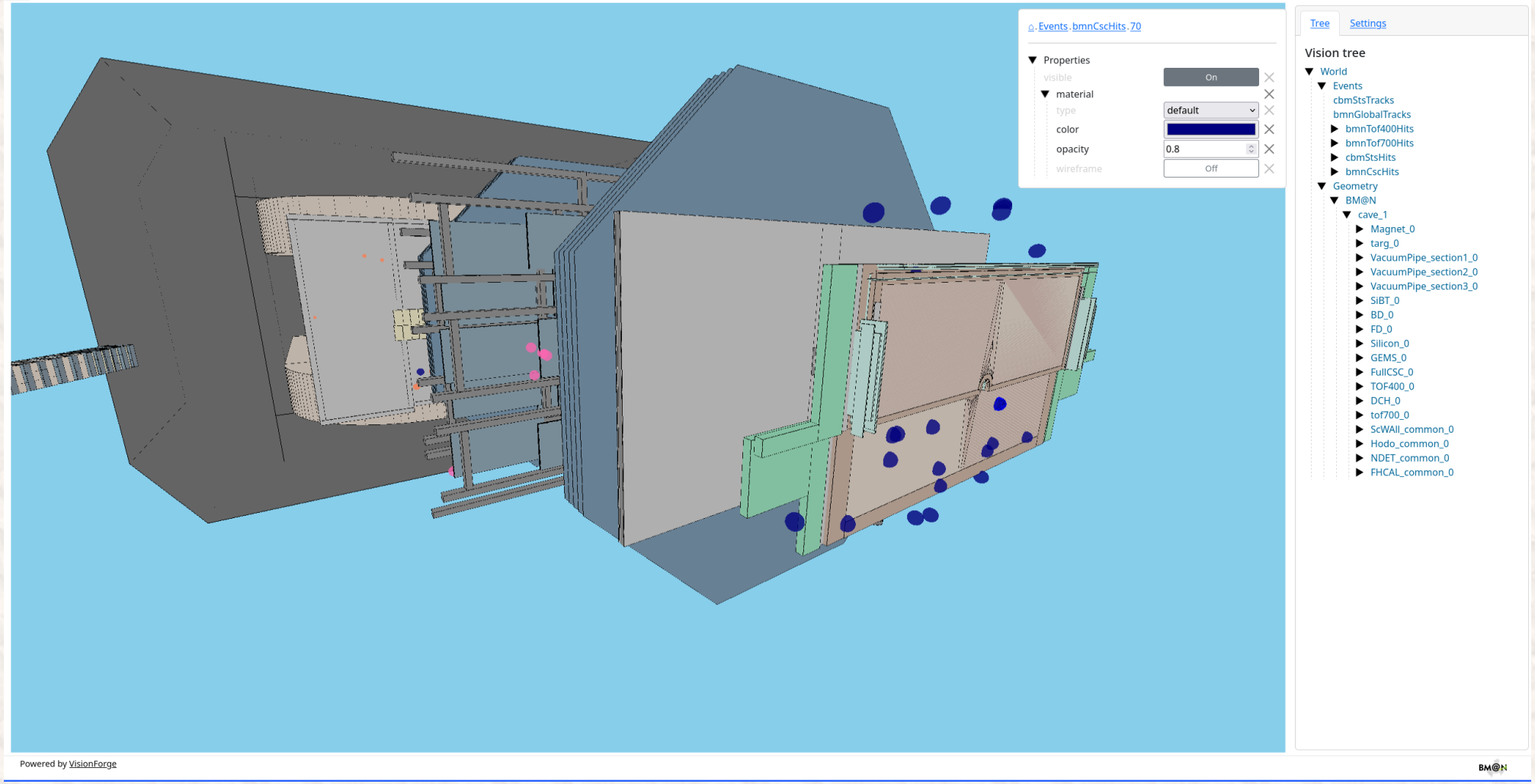


- Reuses most of the previously implemented modules;
- Depends on ROOT for geometry conversion;
- Multiple input formats (JSON, compressed JSON, Protobuf);
- Support for WebSocket subscription and HTTP polling;
- Uses reactive data flow and RSocket protocol for update propagation.

Examples


Events . bmnTof700Hits . 12
Properties
visible
On
material
type
default
color
opacity
0.8
wireframe
Off
Tree
Settings
Vision tree
World
Geometry
BM@N
cave_1
Magnet_0
MWPC_0
Silicon_0
GEMS_0
CSC_0
TOF400_0
tof700_0
DCH_0
ZDC_common_0
SRCArmTriggers_0
Events
cbmStsTracks
bmnGlobalTracks
bmnTof400Hits
bmnTof700Hits
cbmStsHits
bmnCscHits
Powered by VisionForge
BM@N

Examples


⌂ . Events . bmnCscHits . 70
Properties
visible
On
material
type
default
color
opacity
0.8
wireframe
Off
Tree
Settings
Vision tree
World
Events
cbmStsTracks
bmnGlobalTracks
bmnTof400Hits
bmnTof700Hits
cbmStsHits
bmnCscHits
Geometry
BM@N
cave_1
Magnet_0
targ_0
VacuumPipe_section1_0
VacuumPipe_section2_0
VacuumPipe_section3_0
SiBT_0
BD_0
FD_0
Silicon_0
GEMS_0
FullCSC_0
TOF400_0
DCH_0
tof700_0
ScWAll_common_0
Hodo_common_0
NDET_common_0
FHCAL_common_0
Powered by VisionForge
BM@N



Work in progress

- Online monitoring soluiton: the present visulization solution using WebSocket / HTTP to receive data for visualization and to update the data being visualized by web clients via RSocket protocol.
- Saving an image, saving scene parameters (without event or geometry data itself) to a file.
- Visualization of activated calorimeter towers.
- Configurable filters of displayed event objects metadata: particles by their code, energy range, only primary tracks.
- Show general information: selected setup geometry, event number, number of events (if from file), number of displayed geometry objects.
- Virtual reality functionality

Thank You!