



# Software contribution from MIPT: Development of software systems for BM@N

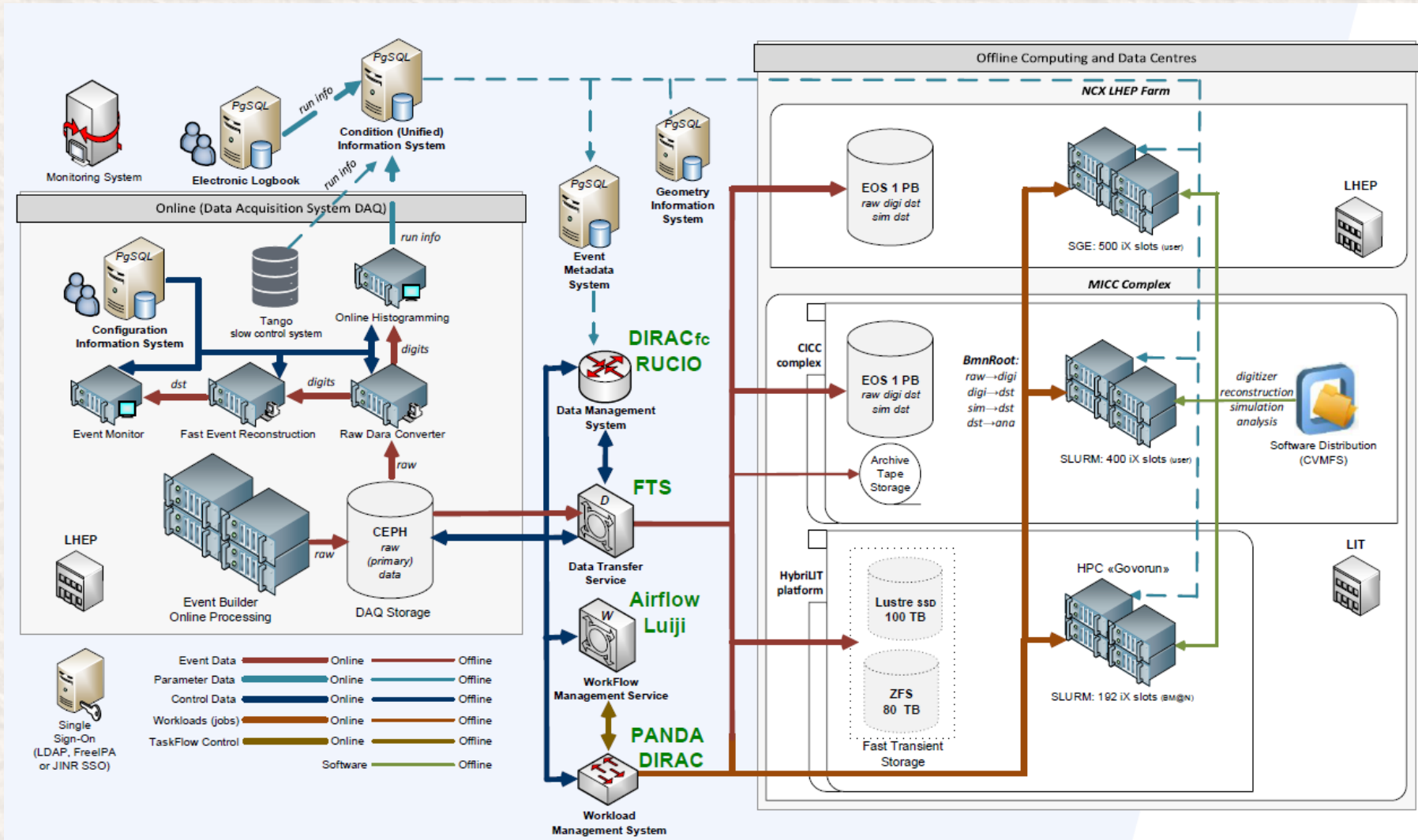
Peter Klimai <[pklimai@gmail.com](mailto:pklimai@gmail.com)>

for the MIPT team





# BM@N Software High-Level View





# Recent Projects Summary

---

| Project  | URL   |
|--|---|
| Development of Event Metadata System components, incl. automated recovery and automated deployment solutions                   | <a href="https://git.jinr.ru/nica_db/emd">https://git.jinr.ru/nica_db/emd</a><br><a href="https://git.jinr.ru/pklimai/ems-stat-collector">https://git.jinr.ru/pklimai/ems-stat-collector</a><br><a href="https://git.jinr.ru/pklimai/ems-deploy">https://git.jinr.ru/pklimai/ems-deploy</a> |
| Development of a service for monitoring software systems of the BM@N experiment  | <a href="https://git.jinr.ru/pklimai/mon-service-deploy">https://git.jinr.ru/pklimai/mon-service-deploy</a>   |
| Development of next-generation event display solution for BM@N, incl. reading ROOT files with event data and detector geometry | <a href="https://git.jinr.ru/idunaev/visionforge">https://git.jinr.ru/idunaev/visionforge</a><br><a href="https://git.jinr.ru/pklimai/visapi">https://git.jinr.ru/pklimai/visapi</a>  |
| Development of REST API service for slow control system  | WIP<br>See also <a href="https://bmn-tango.jinr.ru">https://bmn-tango.jinr.ru</a>   |



# MIPT Software for BM@N Team

Supervision: T. A.-Kh. Aushev

Team members:

- P. Klimai
- A. Nozik
- I. Dunaev (student 5y)
- E. Blinova (student 3y)
- O. Nemova (student 5y)
- A. Degtyarev (PhD st. 1y)
- S. Efimov (student 6y)

The collage displays three main software interfaces:

- BM@N Event Metadata System:** A dashboard showing event metadata with search filters, a total count of 50,000 events, and two pie charts labeled 'My Stat Graph TWO-1' and 'My Stat Graph TWO-2'.
- BM@N Slow Control Viewer:** A window titled 'Tango Parameter' showing a line graph of 'mpd/dag/runcontrol/ev\_number' over time (Apr 2, 2018). It includes a 'Run Selector' with start and end times.
- BM@N EVIS:** A 3D visualization of a detector structure with various components labeled in a tree view on the right. It includes a 'Properties' panel and a 'Color' selection tool.

At the bottom, an 'ERROR PANEL' displays a database error: 'DETAIL: Key (detector\_name)=(qq) already exists.' and a corresponding SQL INSERT statement.



# Development of Next-Generation Event Visualization Platform for BM@N



# VisionForge Project

---

See also: [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)  
[https://www.youtube.com/watch?v=uT5j-xOXC3E&ab\\_channel=JPoint%2CJoker%D0%B8JUGru](https://www.youtube.com/watch?v=uT5j-xOXC3E&ab_channel=JPoint%2CJoker%D0%B8JUGru)

- 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
  - Using Kotlin-Multiplatform



# Geometry, tracks, scene graph, tuning

Menu **BM@N EVIS** Sign In

Period number: 8 Run number: 8000 File address: /eos/bmn/8-357.example Fetch Select event: 1250

△.BM@N.cave\_1.DCH\_0.DCHDetV\_0.  
DCHCoverLayerV\_0.OctagonCoverPlaneS

▼ Properties

visible

▼ material

type default

color

opacity 1

wireframe

Tree Settings

Vision tree

- ▼ World
  - ▼ BM@N
    - ▼ cave\_1
      - ▶ Magnet\_0
        - ▶ Coll\_1
        - ▶ Coll\_2
        - ▶ Pole\_1
        - ▶ Pole\_2
        - ▶ Yoke\_0
      - ▶ targ\_0
      - ▶ VacuumPipe\_section1\_0
      - ▶ VacuumPipe\_section2\_0
      - ▶ VacuumPipe\_section3\_0
      - ▼ SIBT\_0
        - ▶ station0\_0
        - ▶ station1\_0
        - ▶ station2\_0
      - ▶ BD\_0
      - ▼ FD\_0
        - ▶ Qu\_ActiveVolumeV\_1
        - ▶ Prnt\_tube\_mat\_1
        - ▶ Head\_mat\_1
      - ▶ Silicon\_0
      - ▶ GEMS\_0
      - ▶ FullCSC\_0
      - ▶ TOF400\_0
      - ▼ DCH\_0
        - ▶ DCHDetV\_0
        - ▶ FlangeV\_1
        - ▶ FlangeV\_2
        - ▶ TubeV\_1
        - ▶ DCHDetV\_1
        - ▶ FlangeV\_3
        - ▶ FlangeV\_4
        - ▶ TubeV\_2
        - ▶ tof700\_0
        - ▶ ScWall\_common\_0
        - ▶ Hodo\_common\_0
        - ▶ NDET\_common\_0
        - ▶ FHCAL\_common\_0
    - cbmStsTracks
    - bmnGlobalTracks

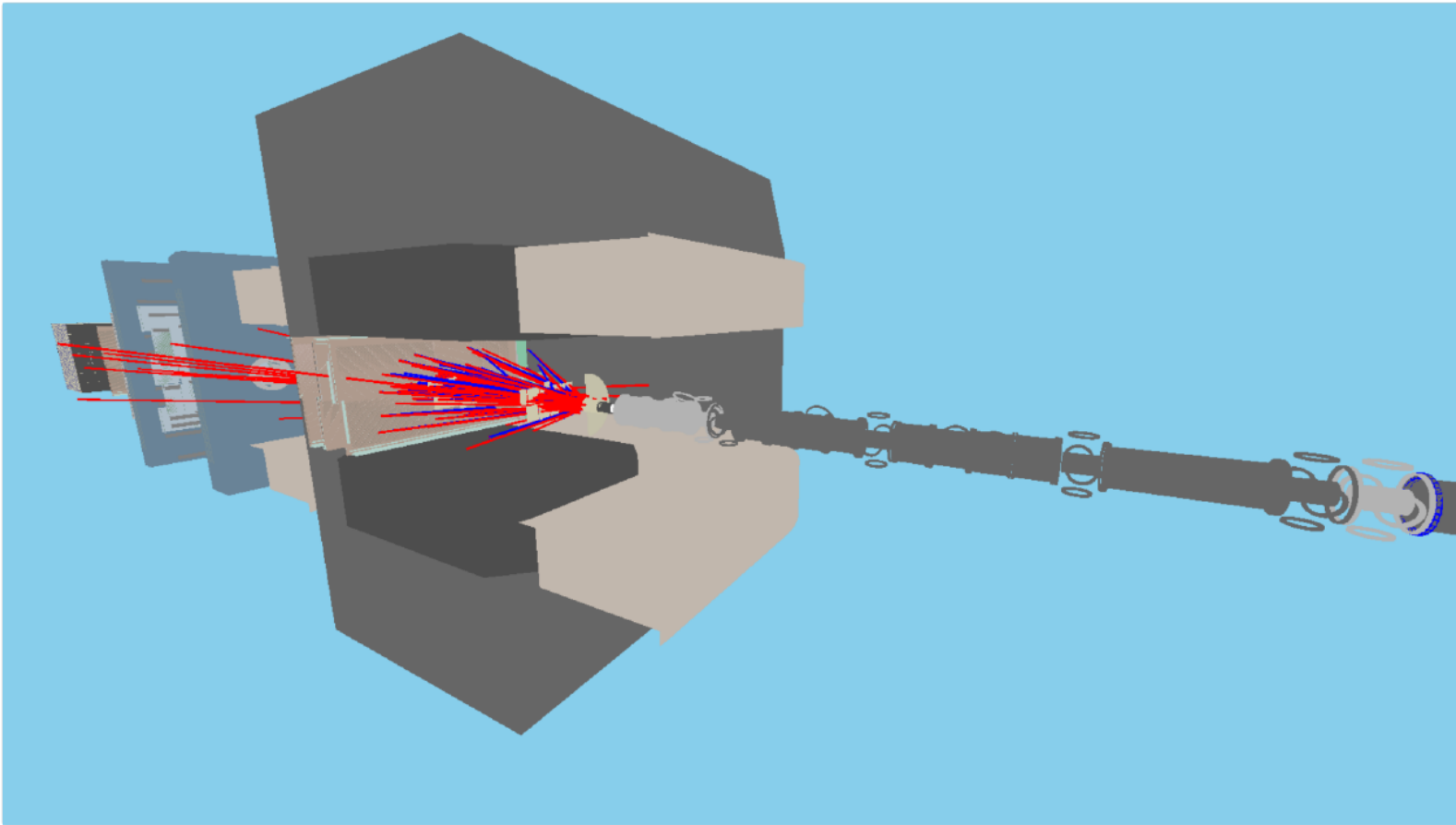
Powered by [VisionForge](#)



# Configuration export example

Menu **BM@N EVIS** Sign In

Period number:  Run number:  File address:   Select event:



Tree Settings

Canvas configuration

Export

Properties

- axes
  - visible
  - size
  - width
- layers
  - 0
  - 1
  - 2
  - 3
- dipping
  - x
  - y
  - z

Powered by [VisionForge](#) BM@N





# CERN ROOT integration

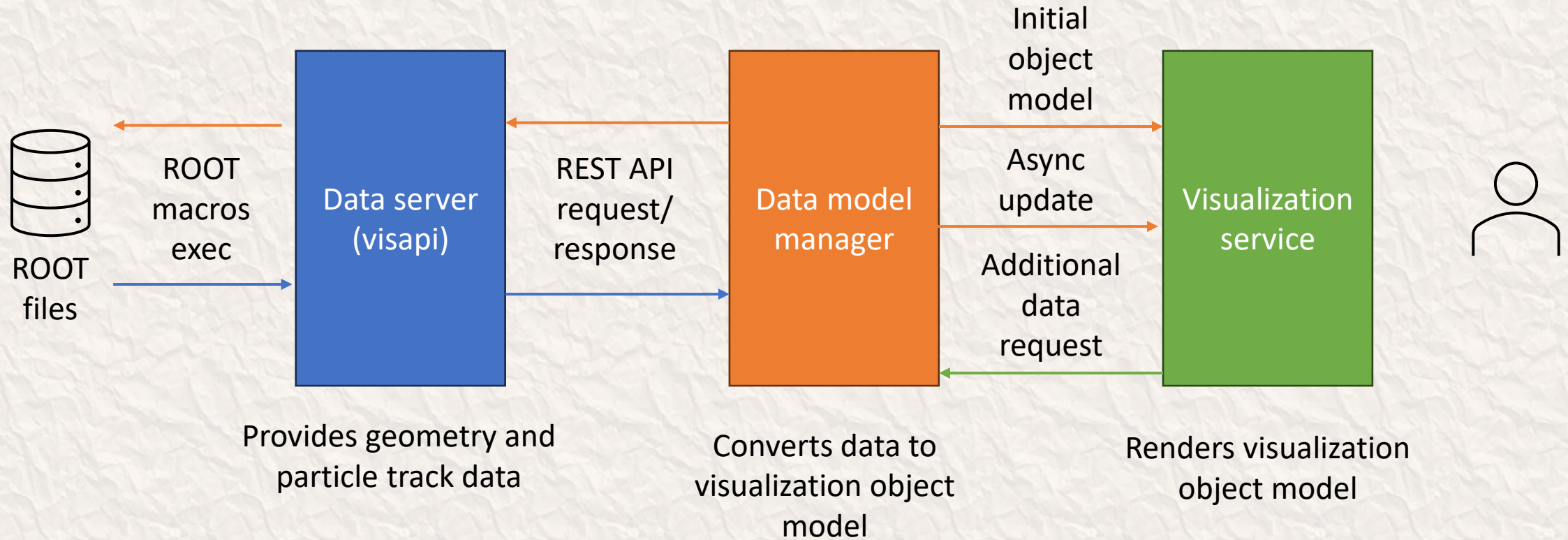
---

## Possible approaches:

- Read ROOT format and convert it to Vision Object Model (**KRootIO** project)
- Create a ROOT plugin that converts TGeoManager to VOM on the ROOT side.
- Convert TGeoManager to JSON via TBufferJSON (**visapi** project)



# General scheme of operation of the VisionForge visualization system when displaying data from ROOT files

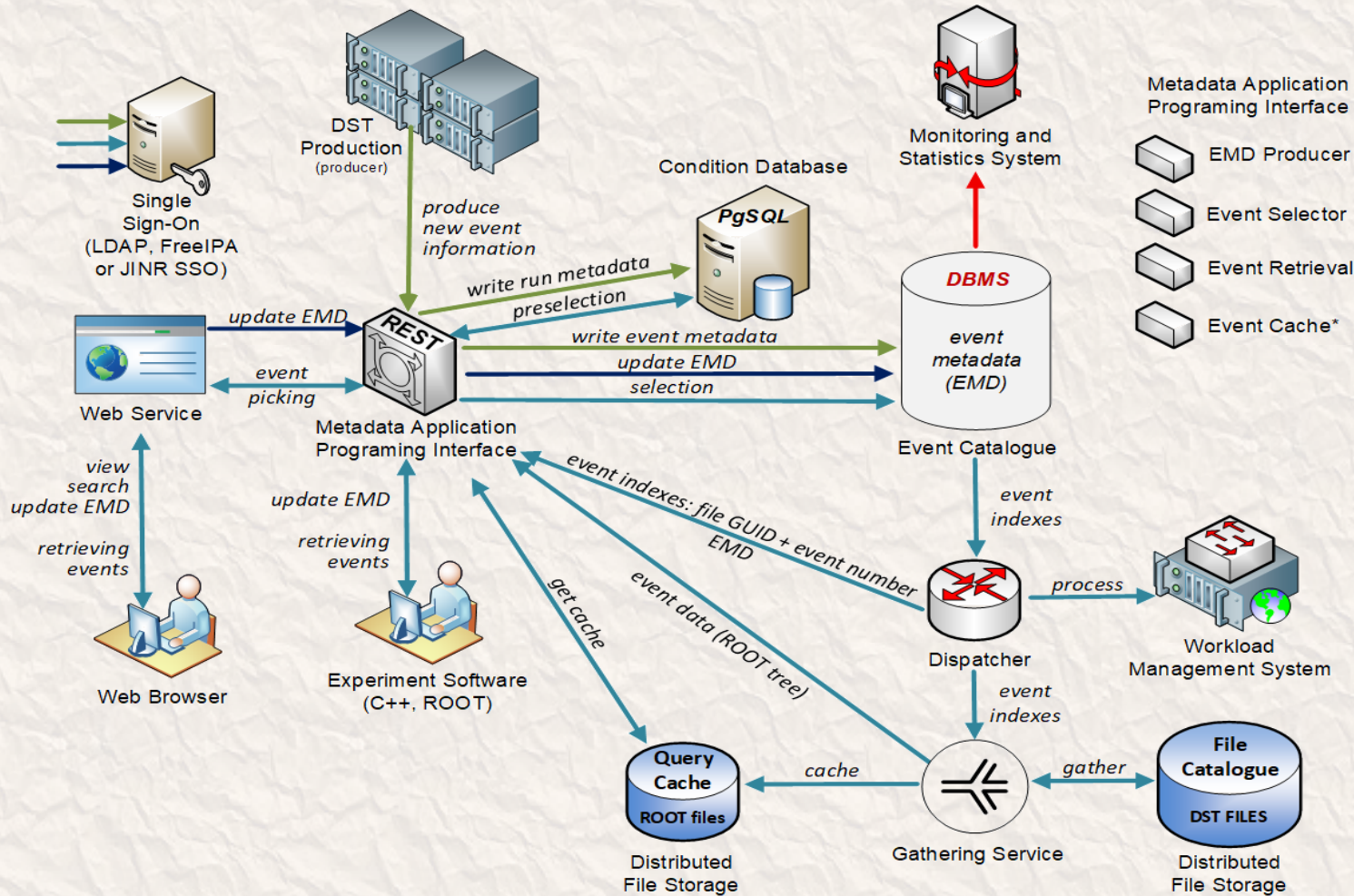




# Development of Event Metadata System components



# BM@N Event Metadata System



## • 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 write BM@N events in the catalogue
- Role-based access control implemented
- 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).



# New REST API scheme for EMS

- The new scheme is unified for different BM@N Information Systems

GET

POST

DELETE

**https://bmn-event.jinr.ru/event\_api/v1/event?**

run\_number=3950:4000&beam\_particle=Ar&target\_particle=Al  
energy=3.16:3.18&target\_particle=SRC%20Lead

HOSTNAME / SERVICE / VERSION / ENTITY?parameter\_set

HOSTNAME=https://bmn-[SYSNAME].jinr.ru

SERVICE=[SYSNAME]\_api

VERSION=v1 (v2...)

ENTITY=tablename without last '\_' (if present)

parameters are separated by '&'  
ranges: min:max → >=min AND <=max  
min: → >=min     :max → <=max

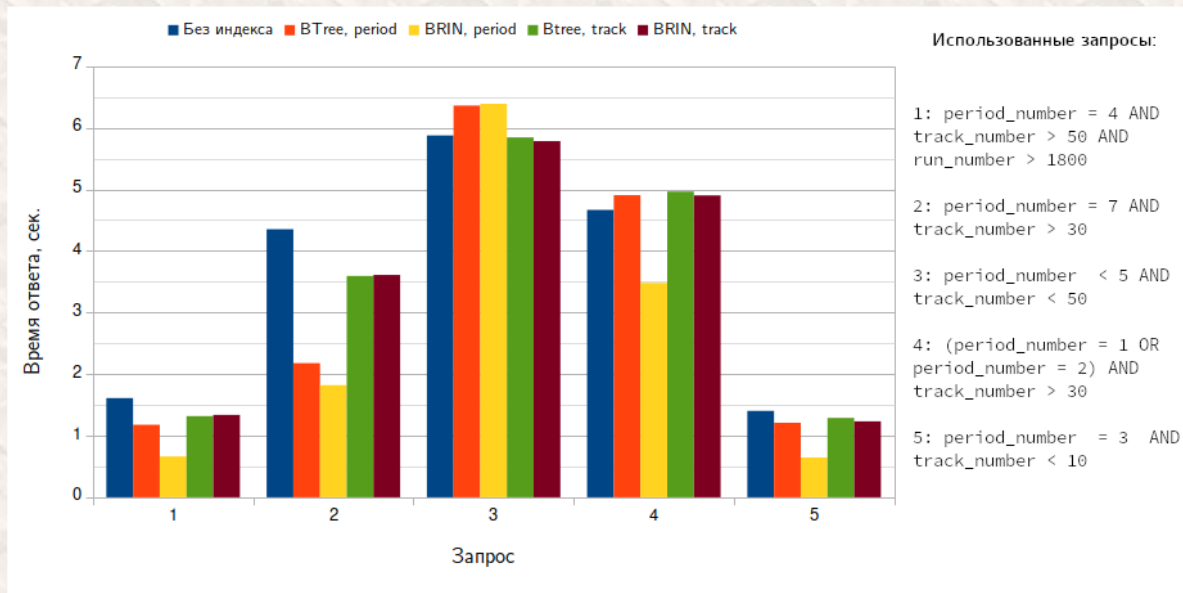
For the Unified Condition Database (UniConDa), SYSNAME = uniconda

For the Event Metadata System (EMS), SYSNAME = event



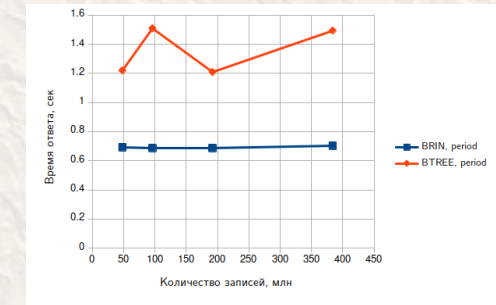
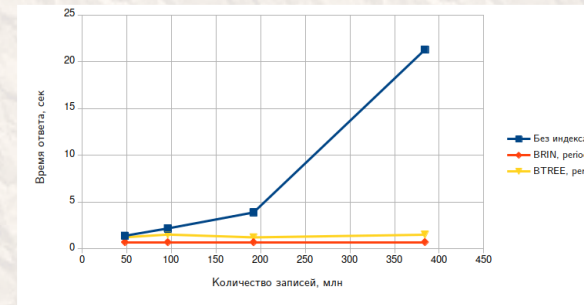
# Event Catalogue Database Optimization

- Measurements with test database instance are shown (50M events)

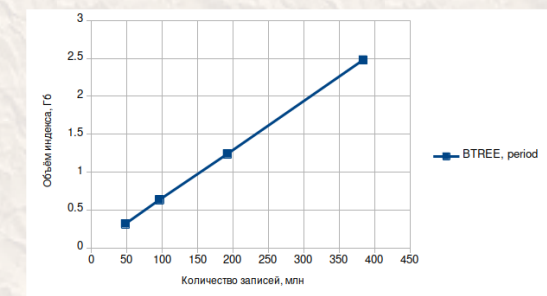
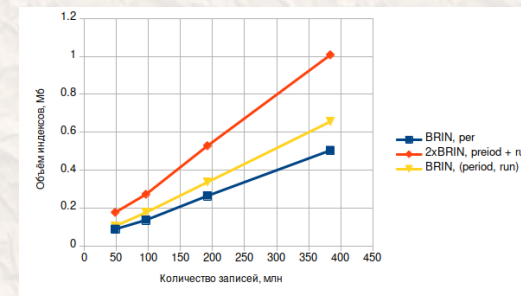


## BRIN vs. BTREE

- Overall, BRIN (Block Range Index) works better for indexing columns having some natural correlation with their physical location within the table



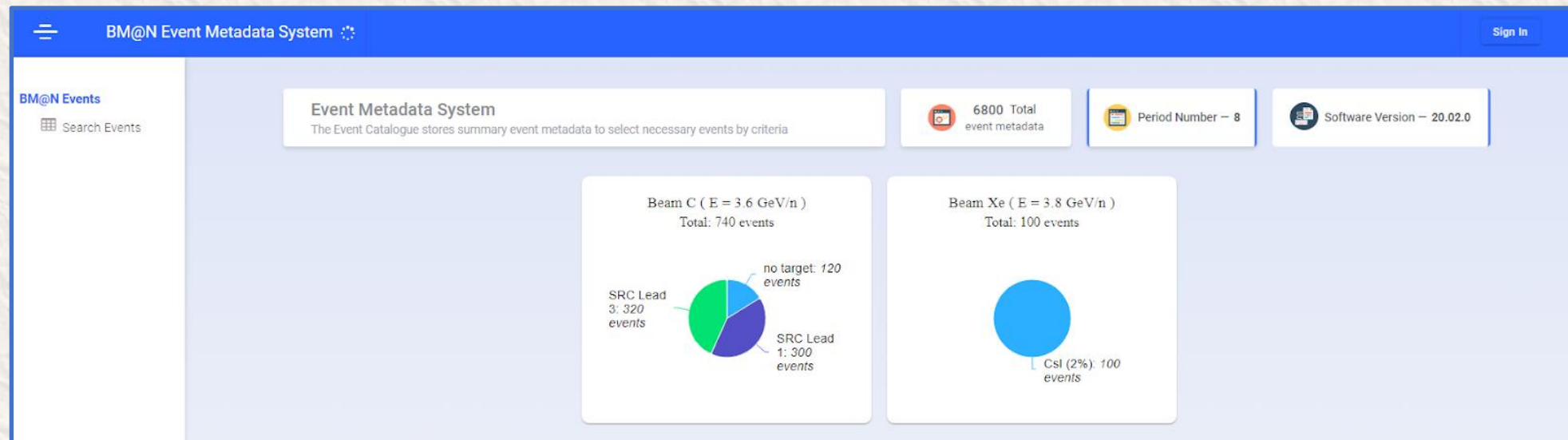
- Adding more periods to test database



- Размеры индексов на диске

# Statistics Collection

- Statistics collection script for EMS Event Catalogue
  - Statistics is saved to database and displayed in Web-interface
  - Script run to be scheduled periodically





# High Availability – Task

---

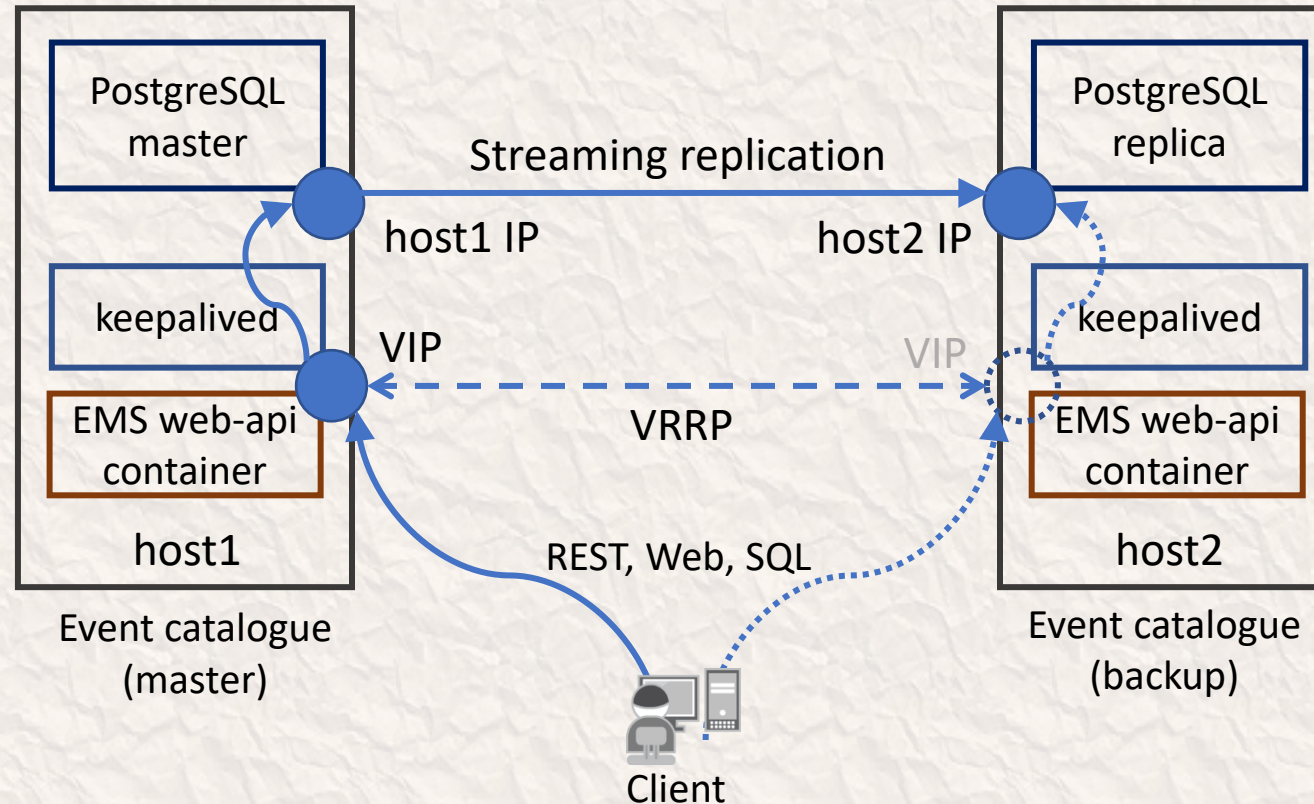
- Need for HA
  - EMS as well as other IS are essential for timely obtaining physical results of the experiment
  - From client point of view, connection must be initiated to single IP / domain name
    - We do not want to ask client to keep several addresses like primary/secondary ones
  - Considering 2 to 1, active/passive redundancy
  - Need to avoid split brain and no brain scenarios





# HA and Automatic Deployment for EMS

- After running Ansible playbooks:





# Development of a service for monitoring software systems of the BM@N experiment



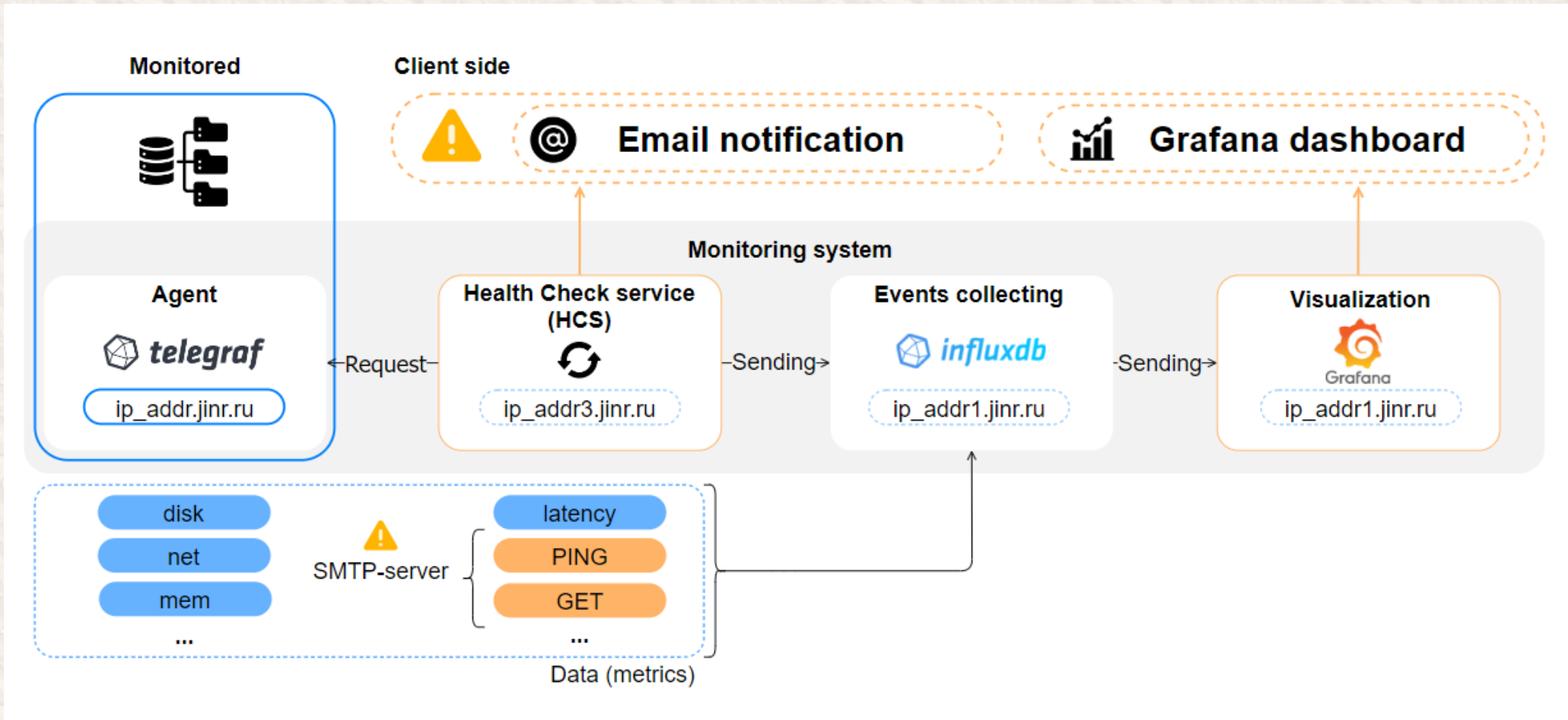
# Monitoring Service – Task

---

- Monitoring Service Features
  - Ping, PG-SQL, or HTTP request to check server reachability
  - Usage of Disk, CPU, Memory, etc. to check health
  - Configurable via JSON file
  - Email notifications
  - Telegraf as host agent
  - InfluxDB for metric storage
  - Grafana for visualization and additional alerting
- Automatic Deployment
  - Ansible-based
  - JSON Dashboards and Alarms auto-generated
  - Software components provisioned automatically



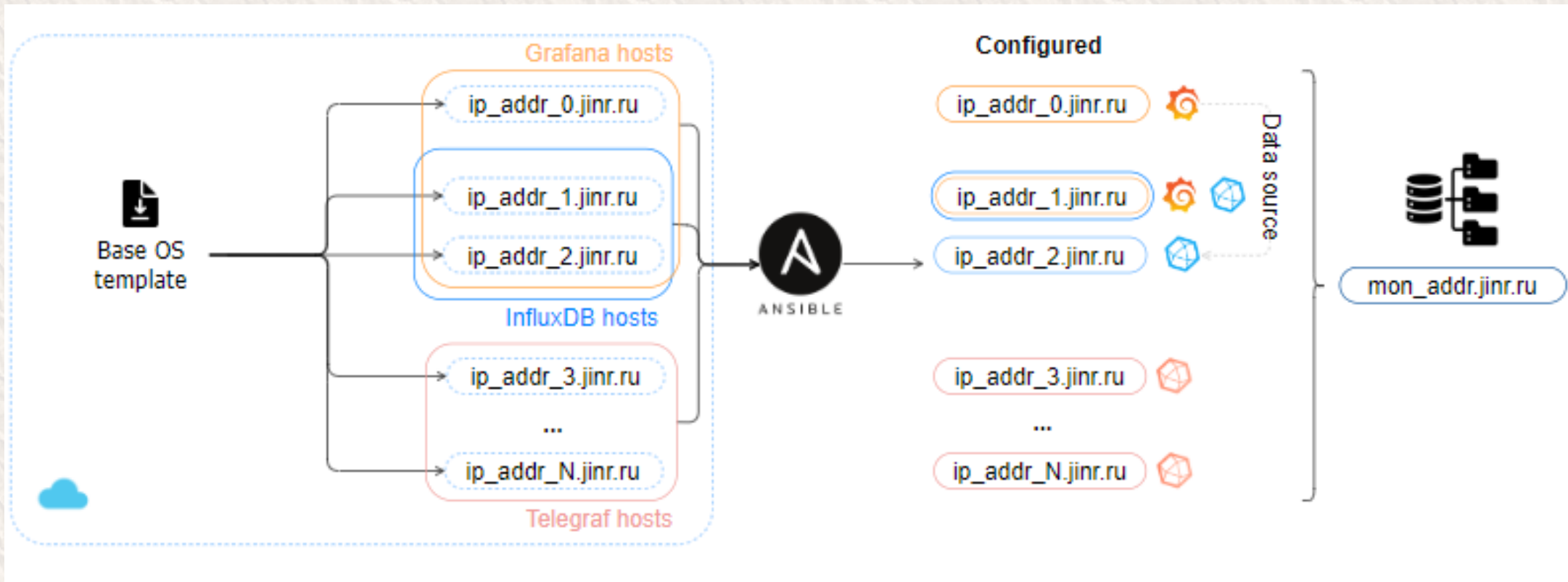
# Monitoring Service Stack and Architecture





# Automatic Deployment

- Ansible Playbooks are used to deploy all service components





# Configuration File Example

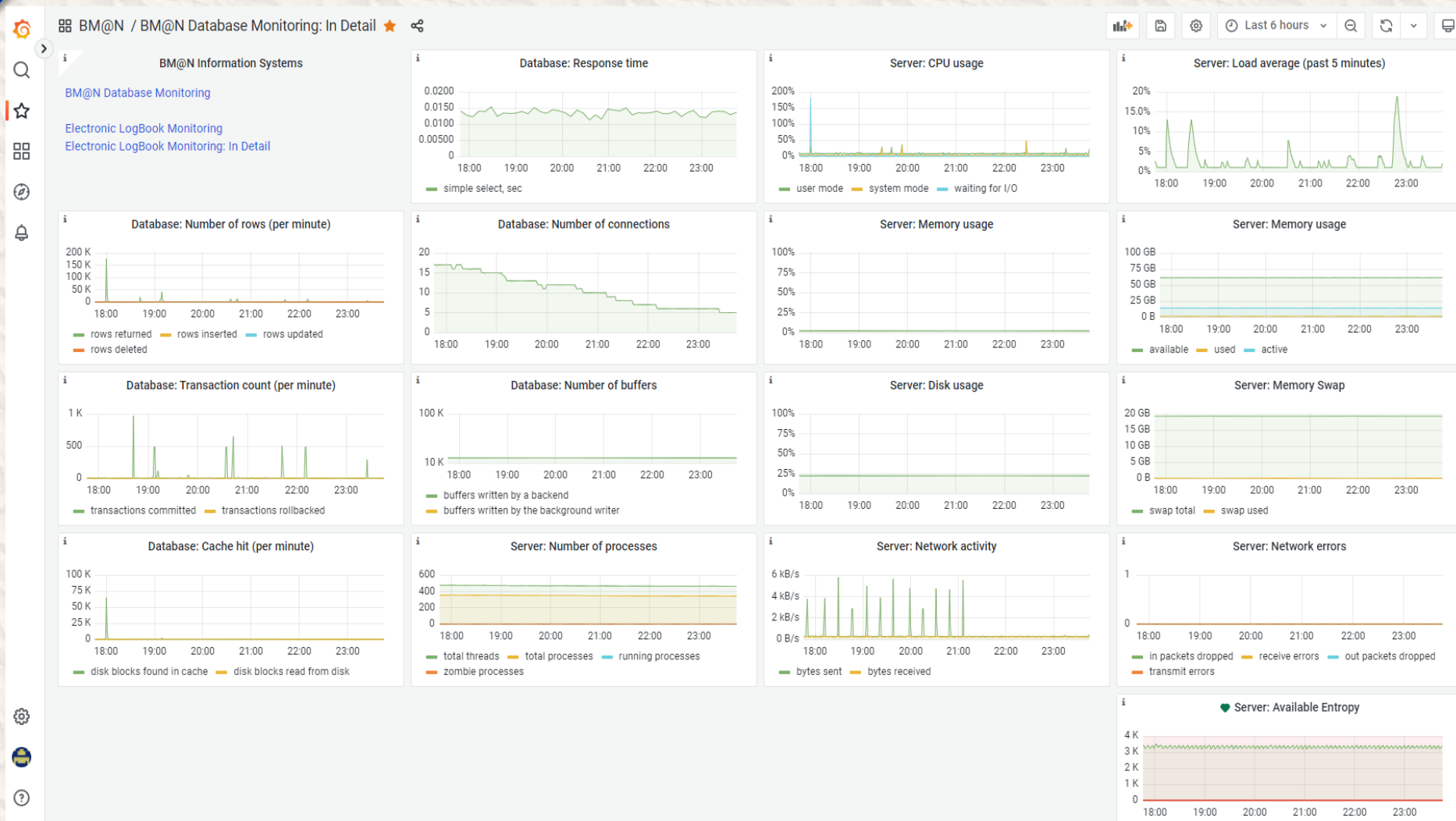
```
{
  "PING": {
    "server1": {
      "IP": "192.168.65.116",
      "NOTIFY": "mail1.jinr.ru"
    },
    "router1": {
      "IP": "10.254.0.41",
      "NOTIFY": "mail2.jinr.ru"
    }
  },
  "DATABASE": {
    "server1": {
      "SERVER": "192.168.65.116",
      "DBMS": "PGSQL",
      "PORT": 5432,
      "DBNAME": "testdb",
      "USER": "testuser",
      "PASS": "****",
      "NOTIFY": "mail3.jinr.ru"
    },
```

```
    "server-centos2": {
      "SERVER": "192.168.65.62",
      "DBMS": "PGSQL",
      "PORT": 5432,
      "DBNAME": "books_store",
      "USER": "bookuser",
      "PASS": "****",
      "NOTIFY":
"mail5.jinr.ru,mail6.jinr.ru"
    }
  },
  "OUTPUT": {
    "DBMS": "INFLUXDB",
    "SERVER": "192.168.65.52",
    "PORT": 8086,
    "DBNAME": "pgsqltest",
    "USER": "influx",
    "PASS": "****",
    "NOTIFY":
"mail1.jinr.ru,mail2.jinr.ru"
  },
```

```
    "INTERVAL_SEC": 60,
    "MAIL": {
      "SERVER": "smtp.yandex.ru",
      "PORT": 587,
      "USER": "****",
      "PASS": "****"
    },
    "LOG": "mail1.jinr.ru,mail2.jinr.ru",
    "NAME": "Monitoring Service"
  }
}
```



# Monitoring Service View Example





Thank You!