



Visualization and auxiliary tools development in BM@N experiment

Peter Klimai



MIPT-NPM Projects for BM@N

MIPT-NPM projects for BM@N:

- Visualization service event display
- Monitoring service
- NICA-Scheduler multiplatform GUI
- Slow control viewer

The work is supervised by **Tagir Aushev** from MIPT-HEP laboratory.

About MIPT-NPM:

- <http://npm.mipt.ru/ru/>
 - <https://research.jetbrains.org/groups/npm>
-



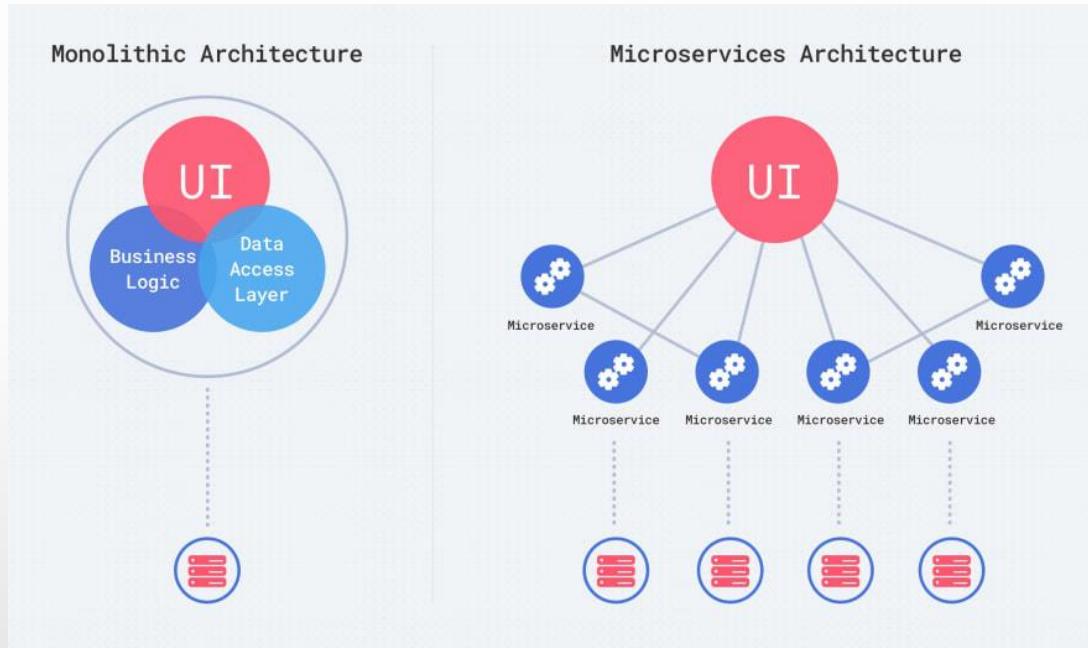
Sponsored by



Visualization – Event Display Next-Gen



Service-based architecture

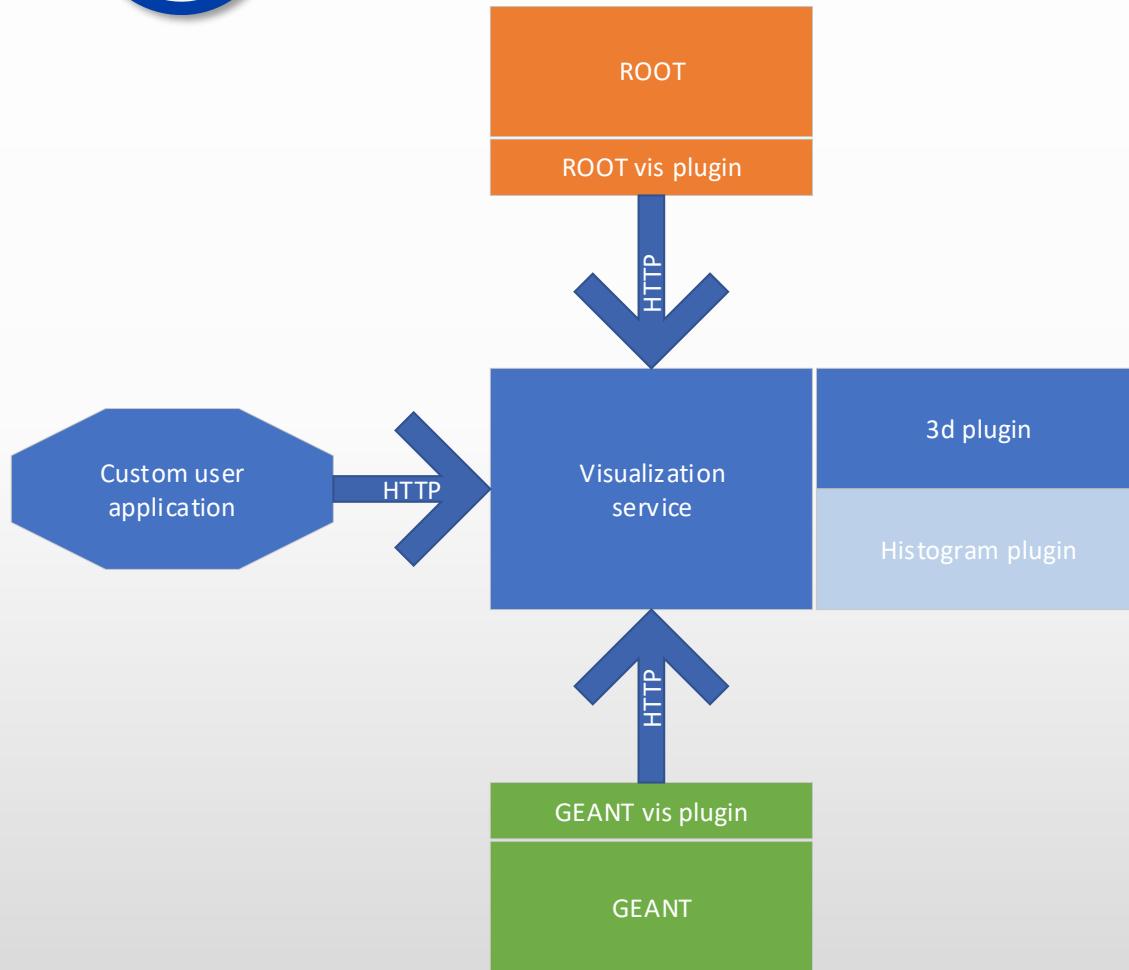


https://dev.to/alex_barashkov/microservices-vs-monolith-architecture-4l1m

- Harder to implement, but much easier to maintain
- Services could be developed or replaced without changing core functionality
- Fast service development cycle
- Easy to move to new infrastructure or scale



Visualization as a service



- Visualization runs as a stand-alone service
- It communicates with other services via HTTP
- The service itself uses plugin system to include new visualization types
- Adapters are made to convert ROOT/GEANT/whatever format into visualization tree



Introducing DataForge-VIS

- Developed as Kotlin multiplatform application
- Available at <https://github.com/mipt-npm/dataforge-vis>
- DataForge-VIS features
 - Working Three-JS rendering engine
 - Caching and template usage for performance optimization
 - Styling capabilities
 - Layering
 - Implementation for all primitives needed for BM@N geometry
 - Basic online object manipulation capabilities
 - Complete reader for GDML format

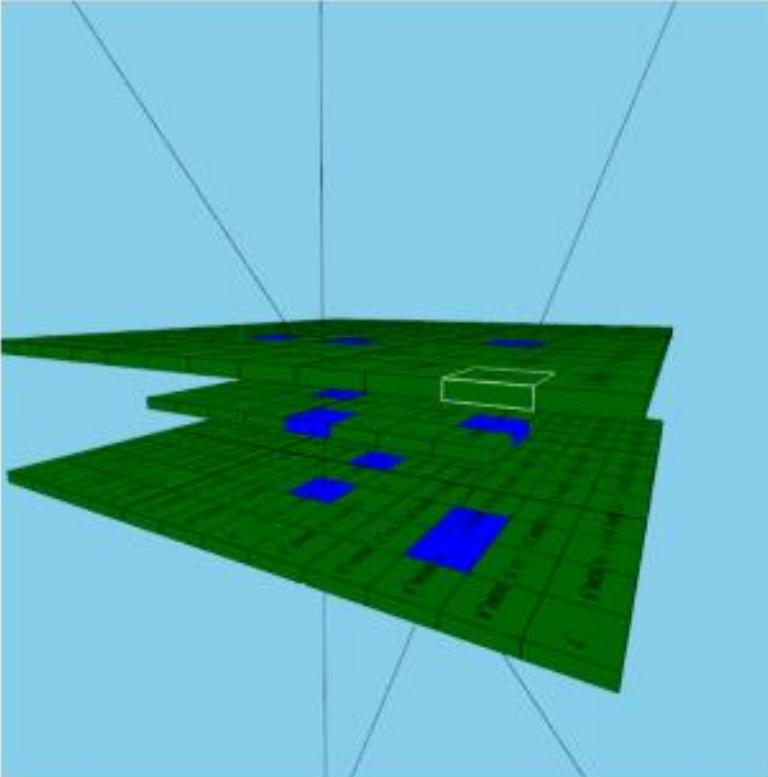


Muon Monitor Demo

Muon monitor demo

Object tree

- ▼ World
 - ▼ top
 - SC72
 - ▼ SC73
 - ◆ SC73_0
 - ◆ SC73_1
 - ◆ SC73_2
 - ◆ SC73_3
 - ◆ SC73_4
 - ◆ SC73_5
 - ◆ SC73_6
 - ◆ SC73_7
 - ◆ SC73_8
 - ◆ SC73_9
 - ◆ SC73_10
 - ◆ SC73_11
 - ◆ SC73_12
 - ◆ SC73_13
 - ◆ SC73_14



Settings

Axes Export

Layers

0 1 2 3 4 5 6 7 8 9 10 11

Events

Next Clear

Properties

top / SC73 / SC73_0

Object > material >

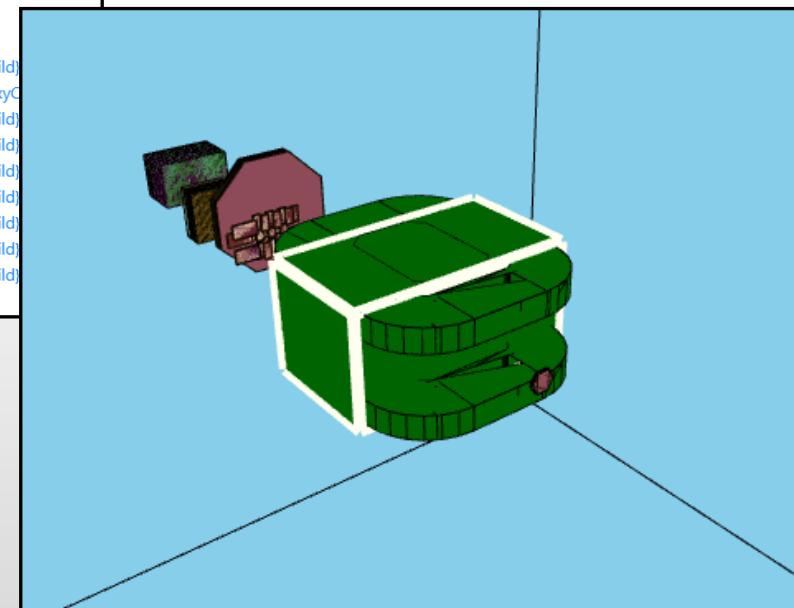
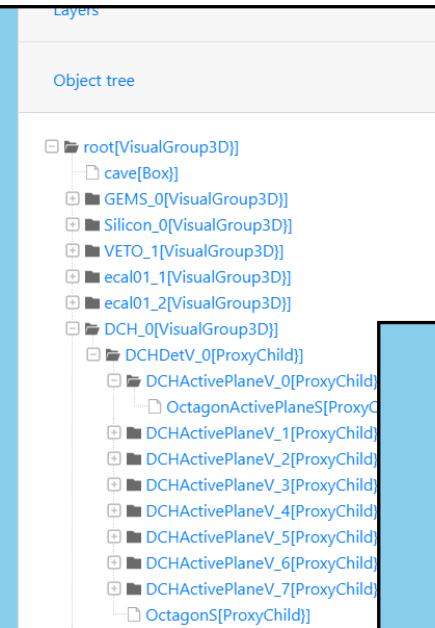
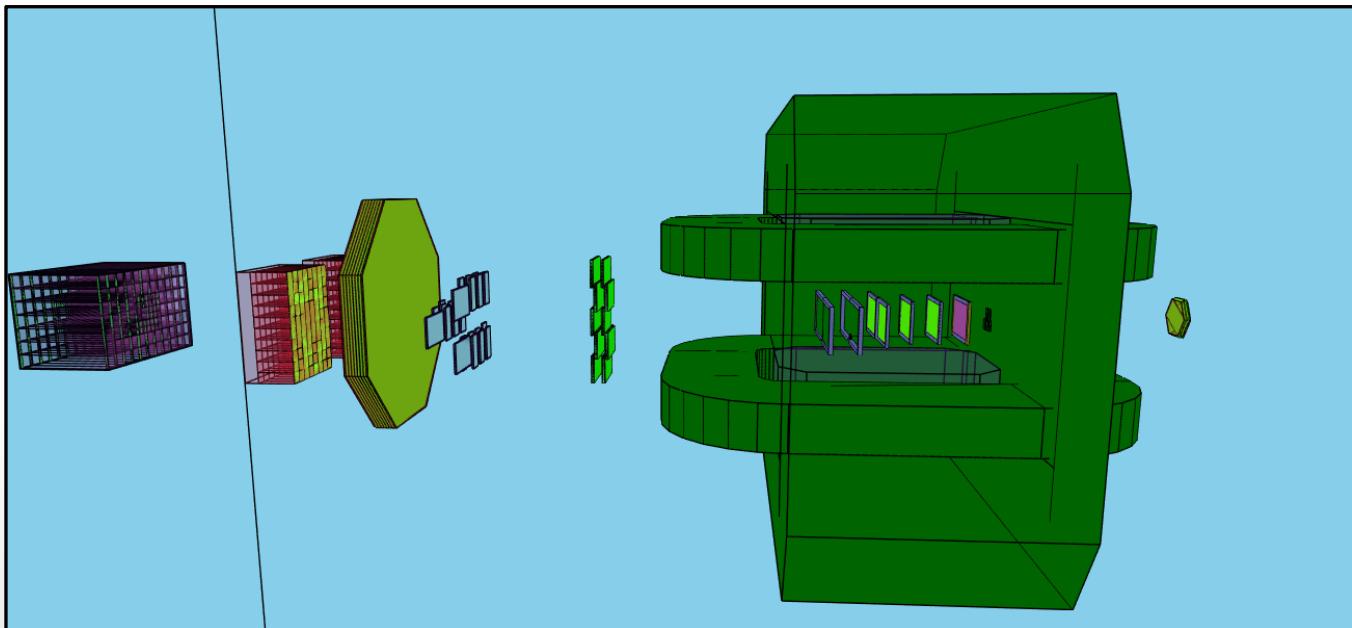
▼ object : {2}

▼ material : {3}

- wireframe : false
- color :
- opacity : 1
- visible : true



BM@N Geometry





Current work and plans

- Finalization of ROOT plugin to convert TGeoManager to JSON
 - <https://github.com/Unrealf1/RootExtraction>
- VR via Three-JS
- OpenGL backend
- Non-3D plugins
- jupyter-kotlin integration (JetBrains internships, summer 2020)



Monitoring Service

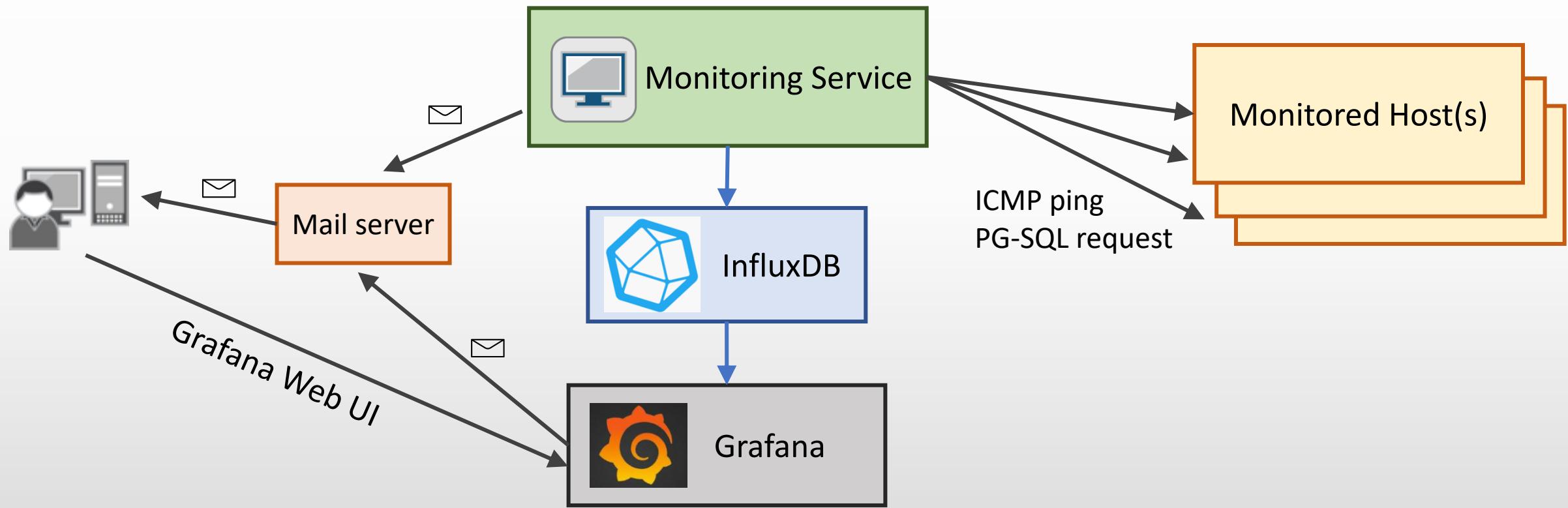


Monitoring Service - Task

- Monitoring Service Features
 - Ping and PG-SQL request to check database status
 - Configurable via JSON file
 - Email notifications
 - Response time stored in InfluxDB
 - Use Grafana for visualization and additional alerting
- Sources:
https://git.jinr.ru/nica/bmnroot/-/tree/dev/uni_db/services/monitoring



Monitoring 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"  
}  
}
```



Grafana View





Email Examples

Screenshot of an email client interface showing a list of messages and a detailed view of one message.

Unread Starred Contact Tags Attachment

Filter these messages <Ctrl+Shift+K>

Subject	Correspondents	Date
[OK] PGSQl response time alert	Grafana	2:41 PM
Service Monitor on CentOS7: server1 - PGSQl state changed to UP	ch@yandex.ru	2:40 PM
[Alerting] PGSQl response time alert	Grafana	2:01 PM
Service Monitor on CentOS7: server1 - PGSQl state changed to *** ...	ch@yandex.ru	1:54 PM

From Grafana <ch@yandex.ru> ☆

Subject **[OK] PGSQl response time alert**

To Me ☆

[OK] PGSQl response time alert

Grafana: Database monitoring warning!

PGSQL response time

0.12 1.00



Scheduler-GUI



NICA Scheduler Configurator: Task

NICA-Scheduler is a module of BmnRoot software

- Uses existing batch system (SLURM, SGE, Torque) to distribute user jobs on the cluster
- Jobs for distributed execution are described and passed as XML file:
`$ nica-scheduler my_job.xml`
- Details: <http://bmn.jinr.ru/nica-scheduler/>
- Task: Create a GUI editor for scheduler XML files

NICA-Scheduler

If you have time-consuming tasks, many simple tasks or a lot of files to process, you can use *batch systems* on distributed clusters, such as SLURM on the **HybriLIT platform** including supercomputer Govorun, Sun Grid Engine (SGE) on the NICA **ncx-cluster** or Torque on the JINR **CICC complex** to essentially accelerate your work.

If you know how to work with SLURM ([SLURM on HybriLIT](#)), Sun Grid Engine ([SGE user guide](#)) and Torque systems ([doc index](#)), you can use ***sbatch*** or ***qsub*** commands on the clusters to parallel data processing. Simple examples of user jobs for SLURM, SGE and Torque can be found in BmnRoot [here](#). Otherwise, NICA-Scheduler has been developed to simplify running of user tasks in parallel.

NICA-Scheduler is a module of the BmnRoot software. It uses an existing batch system (SLURM, SGE and Torque are supported) to distribute user jobs on the cluster and simplifies parallel job executing without knowledge of the batch system. Jobs for distributed execution are described and passed to NICA-Scheduler as XML files, e.g.:



Example of NICA-Scheduler Job

```
<job name="reco_job">
  <macro path="$VMCWORKDIR/macro/mpd/reco.C" start_event="0" event_count="1000" add_args="local">
    <file input="$VMCWORKDIR/macro/mpd/evetest1.root" output="$VMCWORKDIR/macro/mpd/mpddst1.root"/>
    <file input="$VMCWORKDIR/macro/mpd/evetest2.root" output="$VMCWORKDIR/macro/mpd/mpddst2.root"/>
    <file sim_input="energy=3,gen=urqmd" output="~/mpdroot/macro/mpd/evetest_{counter}.root"/>
  </macro>
  <run mode="global" count="25" config="~/mpdroot/build/config.sh"/>
</job>
```

It is perfectly fine to create such a file using a text editor.
You have to remember or look up XML tag and attribute names and make sure XML syntax is valid.



NICA-Scheduler-GUI Details

- Written as a Kotlin Multiplatform application
 - Runs as JavaScript in browser, or in JVM
 - Using the same data model defined in Common part
 - Libraries used:
 - xmlutil (<https://github.com/pdvrieze/xmlutil>)
 - Kotlinx-HTML (<https://github.com/Kotlin/kotlinx.html>)
 - Bootstrap (<https://getbootstrap.com/>)
 - TornadoFX (<https://github.com/edvin/tornadofx>)
- Available at https://git.jinr.ru/nica_modules/mpd-scheduler-gui

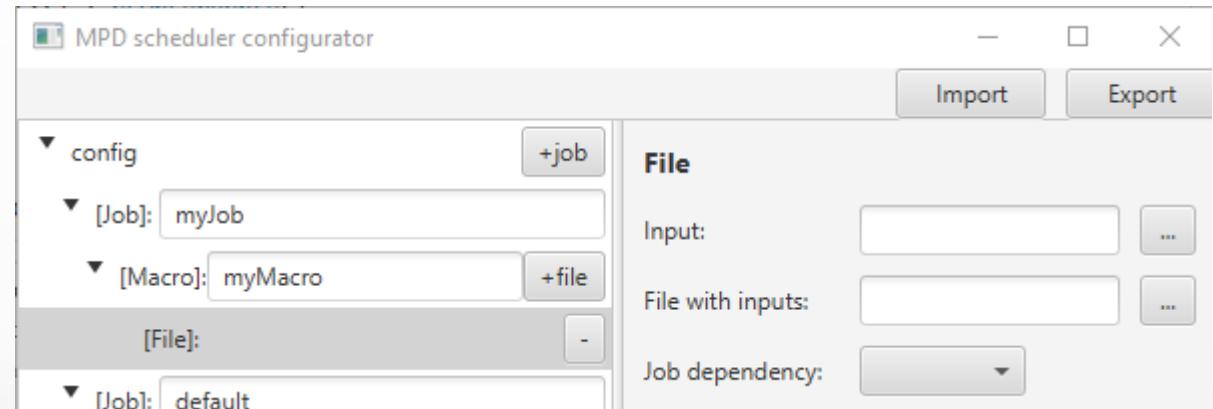


Going multiplatform

JVM

- Cross-platform on desktops
- Easier to develop and maintain
- In future can support: Full execution control
- **Problems with deployment on systems without JVM**

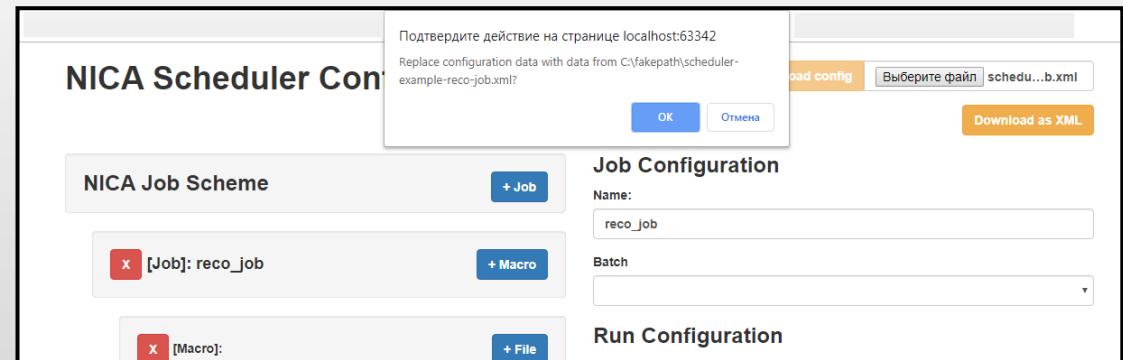
JVM



JavaScript GUI using the same model

- Does not require installation. Could be run from Web site or from local file
- Shares model with desktop via Kotlin-multiplatform technology
- Has almost the same functionality

JS





New design for JS version

NICA Scheduler Configurator

Load config No file chosen

NICA Job Scheme

- [Job]: reco_job
- [Macro]:
- [File]: \$VMCWORKDIR/macro/mpd/evetest1.root
- [File]: \$VMCWORKDIR/macro/mpd/evetest2.root
- [File]: energy=3,gen=urqmd

Files To Process

Input type: Input Criteria:

Output File Path: Start event:

Event count: Parallel mode:

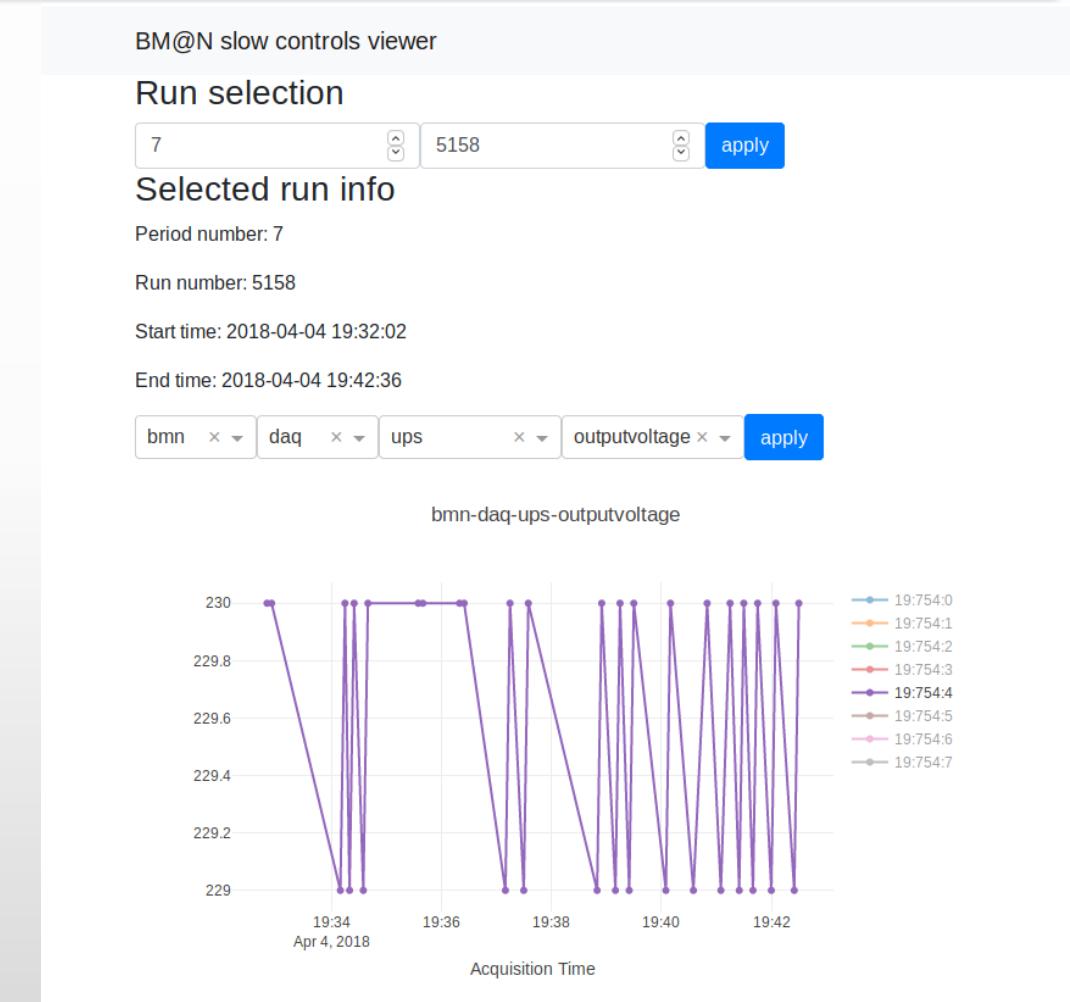
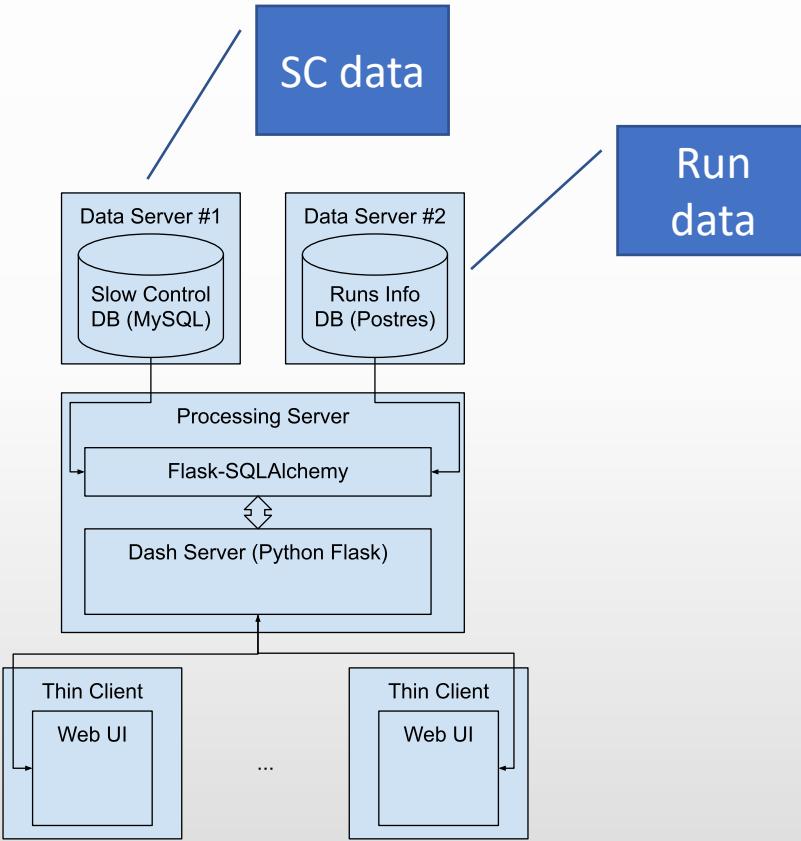
A screenshot of the NICA Scheduler Configurator web application. The interface has a dark header bar with the title 'NICA Scheduler Configurator'. Below the header, there are buttons for 'Load config' and 'Choose File' (with 'No file chosen' displayed), and a red 'Download as XML' button. The main area is divided into two sections: 'NICA Job Scheme' on the left and 'Files To Process' on the right. The 'NICA Job Scheme' section contains a list of selected items: '[Job]: reco_job', '[Macro]', '[File]: \$VMCWORKDIR/macro/mpd/evetest1.root', '[File]: \$VMCWORKDIR/macro/mpd/evetest2.root', and '[File]: energy=3,gen=urqmd'. Each item has a red 'X' icon to its left. To the right of the list are three buttons: '+ Job', '+ Macro', and '+ File'. The 'Files To Process' section contains several input fields: 'Input type' (set to 'Simulation Database'), 'Input Criteria' (containing 'energy=3,gen=urqmd'), 'Output File Path' (containing '~/mpdroot/macro/mpd/e'), 'Start event' (empty), 'Event count' (empty), and 'Parallel mode' (empty). The entire interface is styled with a modern, clean design using dark colors and light text.



Slow Control Viewer



Slow control viewer example





Slow control viewer status

- All components prepared and tested
- Repository available at <https://git.jinr.ru/vchernov/bmn-visualisation>



Thank You!