Web Applications for SPD Computing: CRIC and beyond

Alexey Anisenkov (BINP)

MSCHEP-2026, 13 March 2026

Applications covered in this talk

1. **CRIC** family: A unified Information framework for distributed computing (grid information middleware)
   - JINR (SPD/NICA)
   - CERN experiments (ATLAS, WLCG, CMS ..)
   - DOMA datalake (LSST, sPHENIX)
   - DUNE
2. **SPD Production requests management system**
   - JINR: SPD data processing requests interface
3. **daqweb** family: Web-based control and monitoring platform for DAQ applications
   - CMD-3 experiment (BINP) (generic DAQ monitoring, Online, Nearline, Offline plots and metrics; experiment configuration)
   - Muon G-2 run-1 (Fermilab)
   - MRT X-ray experiments (BINP)

Basic requirements/features for generic conf/web app system

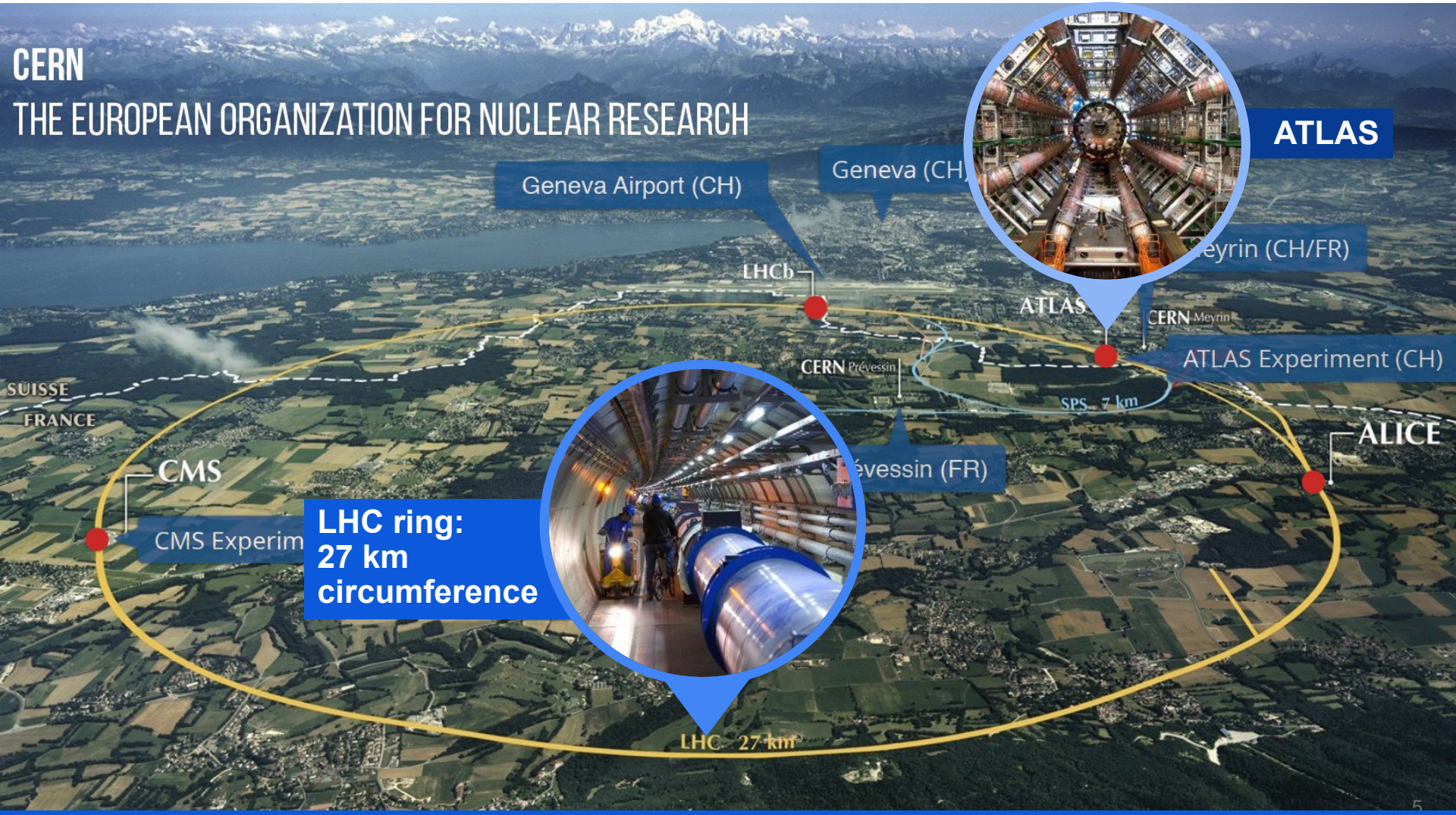
- Implementation of specific Information Model(s) (subject-oriented)

- Built-in data validation & consistency check procedures
- Changes log tracking (history of changes)
- Handy interfaces for data access:
  - API, RESTful export, friendly WebUI, integration widgets, ...
- Access control policy (Permissions Model)
- Usage of auth & authz methods/services (passwd, SSL, PROXY certificates, IAM tokens, etc...)
- Integration of data sources (DBs, internal/external, data collectors)
- Flexible/extensible portal for data modification & browse actions:
  - Update forms with changes confirmation feature
  - Expose data using various structured views (table views, object detailed pages, tree-based views, calendars, cross-navigation, widgets, ... etc)

Overall goal: facilitate successful Operations

CRIC family
(GRID information middleware, LHC Computing)

The Large Hadron Collider (LHC) at CERN



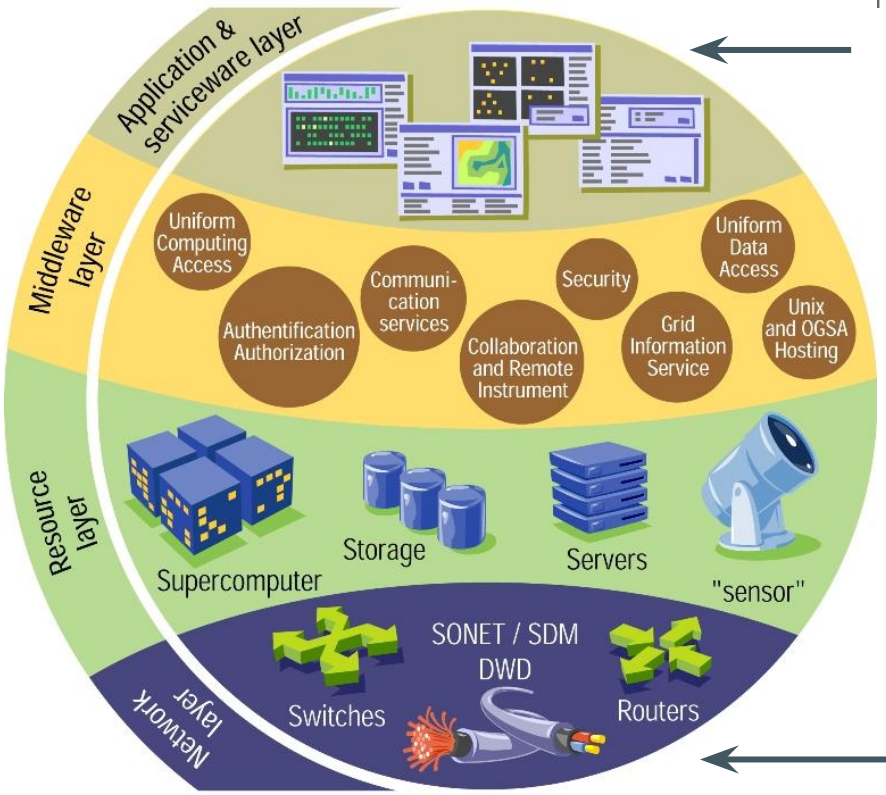
Exploration of new physics and energy frontier in pp and pb-pb collisions

LHC Challenges

- 100+ PB/year capacity production
  - Current total:
    - All LHC: 1.5+ EB
    - ATLAS: 0.5+ EB (raw+sim+derived+replicas)
- Data analysis requires at least ~500k cores (typical PC processor cores)
- Scientists in tens of countries worldwide
- CERN can provide only up to 20-30% of the storage and CPU

Requires powerful large-scale computing & storage system; Distributed-grid concept

Traditional Global Grid Infrastructure layers



High-level VO-oriented middleware services & applications (e.g. for ATLAS: PanDA, Rucio, CRIC, MONIT, TEST frameworks ..)

The middleware exposes heterogeneous resources to VOs in a uniform interface through the Grid:

- Computing Elements give access to CPUs
- Storage Elements give access to data
- Information systems describe the resources
- Authentication & Authorization

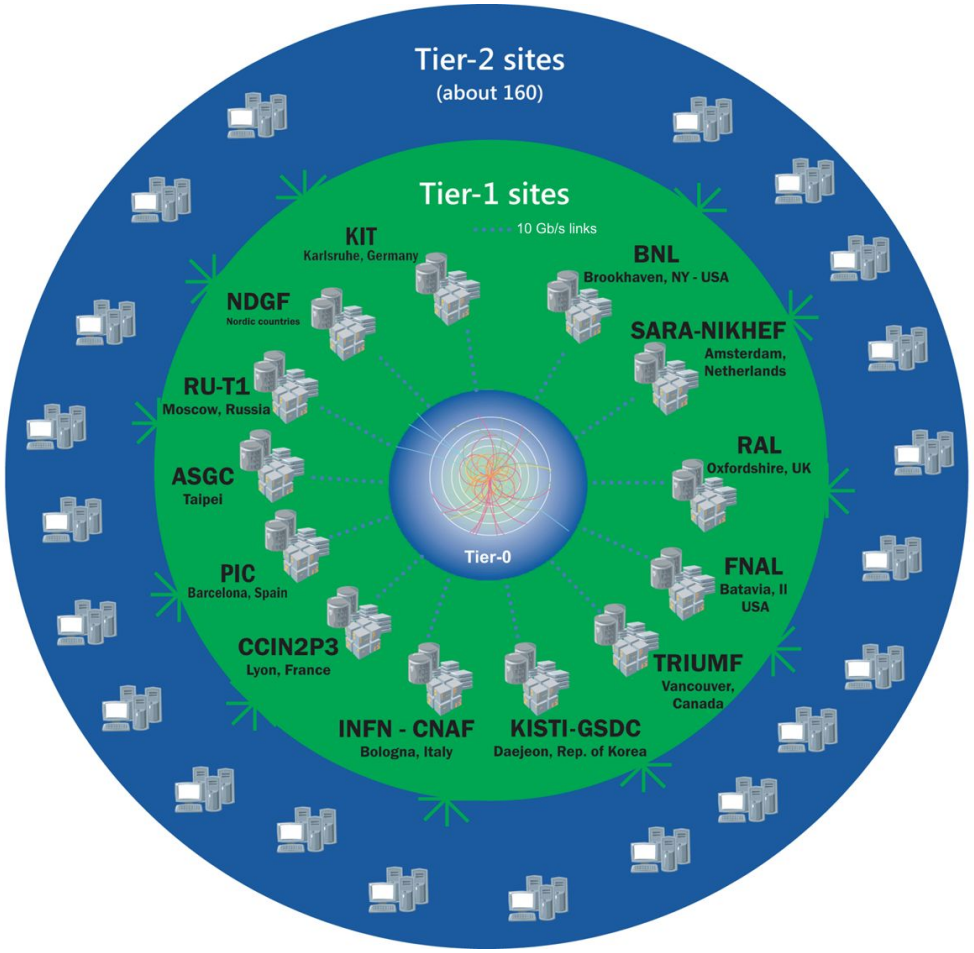
Dedicated LHC optical Private Network

LHCOPN

Middleware makes Illusion that distributed infrastructure is a single resource.

WLCG Computing Model
(continuously evolving)

- **Tier-0 (CERN):**
  data recording and archival, prompt reconstruction, calibration and and distribution
- **Tier-1s:**
  permanent storage, second tape copy of data, re-processing, memory & CPU intensive tasks, analysis
- **Tier-2s + Tier-3s:**
  Simulation, end-user analysis



nearly 170 sites, 42 countries

> 2 million jobs/day
10-100 Gb links (2016)

Pledges resources (2016):

350k cores (3.8M HEPSpec06)

700 PB of storage (310PB disk + 390PB Tape)

Integrates computer centres worldwide that provide computing and storage resource into a single infrastructure accessible by all LHC physicists

* Initial MONARC architecture (1999) - Models of Networked Analysis at Regional Centers for LHC Experiments

Distributed Computing Environment: Resources

~~LHC Experiments~~ (any modern HEP experiments) rely on **heterogeneous** distributed Computing

- variety of computing resources and sub grids involved

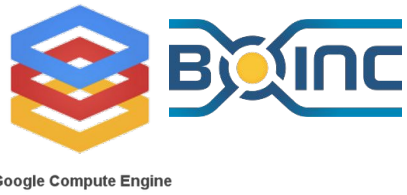
WLCG Pledged resources

Rented, on demand

Opportunistic

Opportunistic backfilling

- variety of infrastructures and middleware providers



others ..

Distributed Computing Environment: Experiments

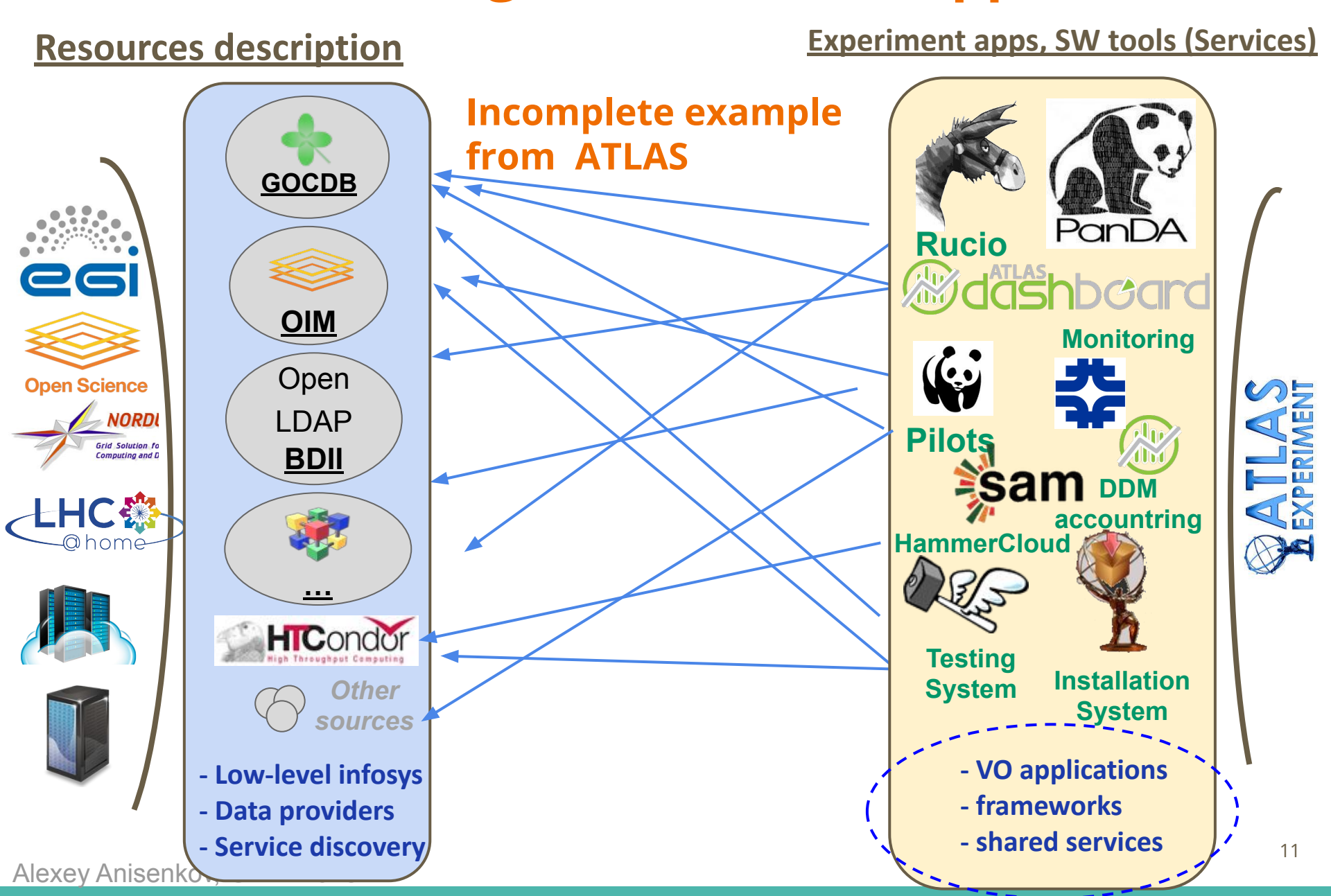
- Each **Community** uses and describes **Resources** in its own way

...

  - Computing Models are similar but still have different implementation
  - Various high level VO-specific frameworks & middleware services (e.g. for Data and Workflow management)
  - Cross experiments applications (monitoring, accounting, testing frameworks, resource usage descriptors, etc)

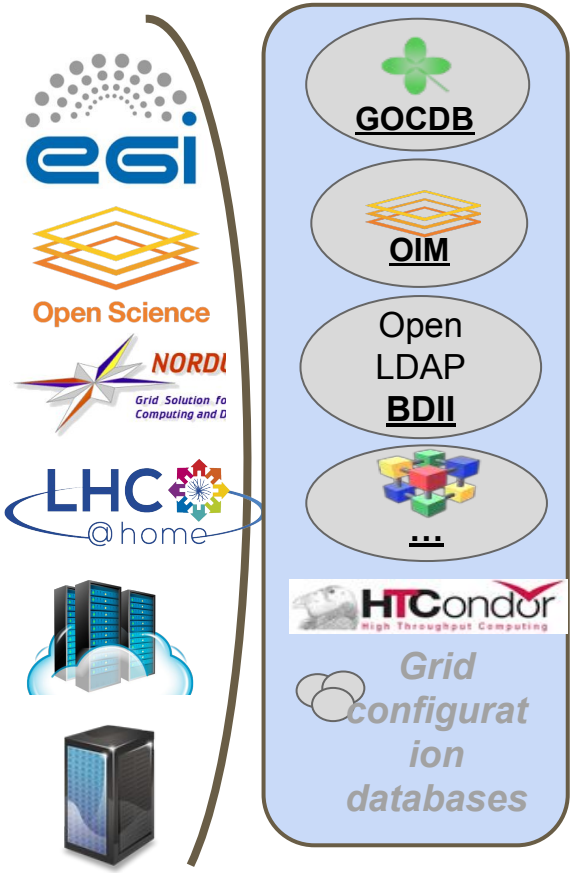
- Apart from **resources** description, high level VO-oriented middleware services and applications also **require** the diversity of common configurations to be centrally stored and shared

Resource Configurations & VO applications



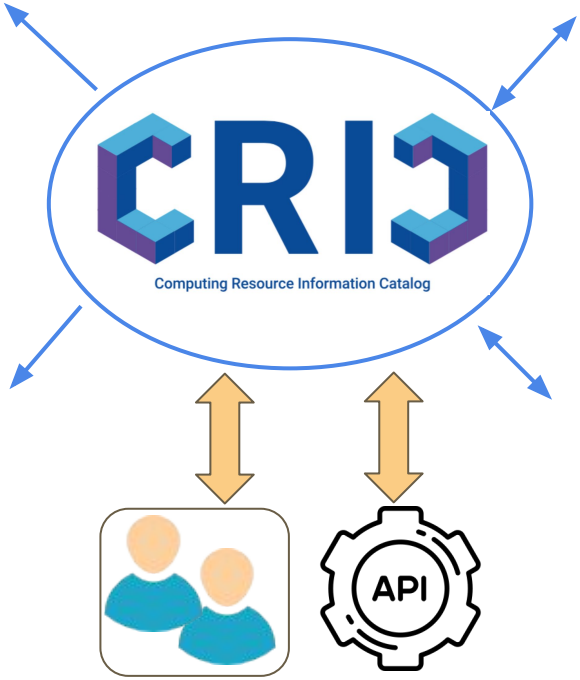
CRIC: a unified configuration system for a large scale, heterogeneous and dynamic computing infrastructure

HW/SW Resources Configuration layer



- Low-level configuration systems
- Data providers
- Service discovery components

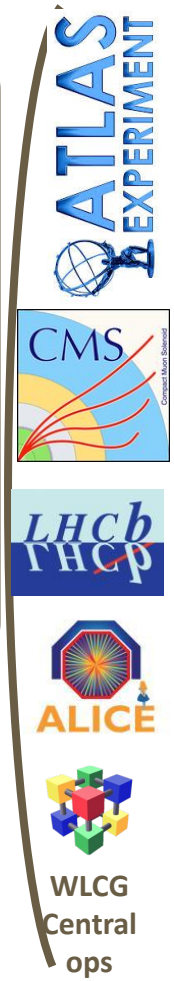
High-level Information middleware



- Central WebUI portal
- REST API services for data export and modification
- Automatic data collectors
- Data validation

Experiment Applications Frameworks, services layer

- Data Management System
- Workload management systems
- Monitoring tools
- SW installation services
- Testing frameworks



Alexey Anisenkov

CRIC mission: link Resources & VOs together

- ➢ Consolidate topology information of a large scale distributed dynamic computing environment
- ➢ Facilitate distributed computing operations for (LHC)** Experiments

Key functional capabilities of the CRIC information concept:

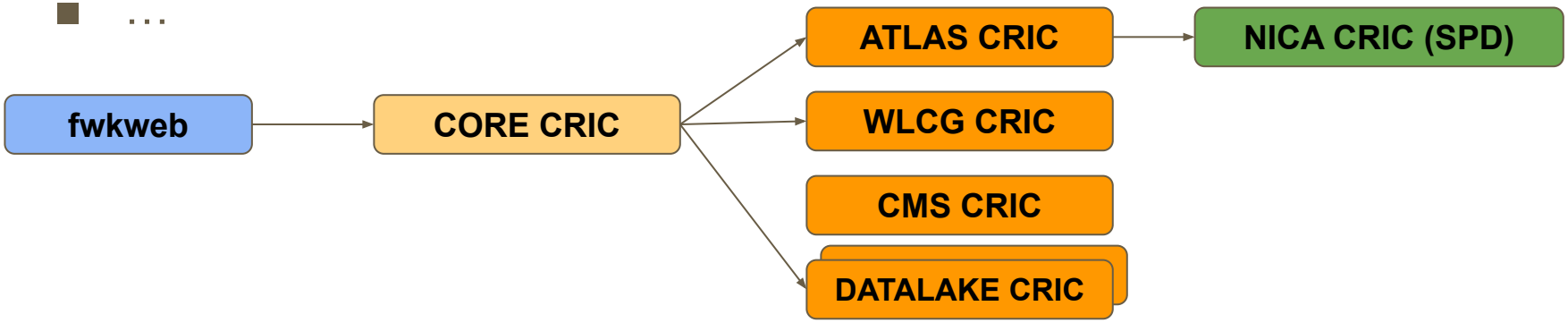
- Built-in information Model(s) for Resource descriptions
- Clear distinction between (physical) resources ***provided by*** grid (Sites) and how they are ***used by*** Experiment(s)
- Built-in aggregation and validation of data collected from various low-level information providers (sources)
- Ability to extend and complement Information Model(s) with Experiment(s) specific data structures. Flexibility to address technology evolution and changes in the VO Computing models and applications
- Experiment-oriented but still Experiment-independent information framework; Plugin based approach allows experiments to address own reqs

** Initially AGIS (CRIC) has been developed for the ATLAS experiment and then evolved to whole LHC computing environment. Today thanks to **Plugin based approach** CRIC can be applied **beyond WLCG** for a generic computing environment as unified information system to address custom VO requirements

CRIC Architecture in a nutshell

- **Plugin based**: VO can configure, extend and customize default behaviour
- **Applications based**: VO enables only needed features out of the box
- VO can rely on other plugins

- Base implementation of Computing model -- GRID specific
- REST API data export (filters, presets, various output formats)
- Shared engine/widgets/apps for WebUI:
  - ➢ calendars, table view, tree view, object view, inline editors, etc..
  - ➢ changes confirmation edit forms
  - ➢ custom object parameters, generic model configurations
- Enhanced Authentication (IAM/SSO, SSL, passwd based; local accounts)
- Enhanced Authorization (instance specific permissions, groups, roles, global actions, map permissions to e-groups, fetch info from ext sources)
- Detailed Changes log history
- …



CRIC family: each plugin activates what is needed

ATLAS CRIC

- ATLAS GRID Topology
- ATLAS Sites configurations (frontier, squids, caches, ..)
- Dynamic configuration (downtimes, blacklisting, Functional tests matrix, ..)
- Compute and Storage configs
- ADC Extra settings/params

WLCG CRIC

- WLCG GRID Topology
- Resource Usage by VOs
- VO pledges and requirements
- Accounting and validations
- WLCG Reports generation tools
- WLCG Resource Review Board (RRB) plots
- TaskForces campaign info

CMS CRIC

- Simplified topology
- Downtime calendars
- Full Users, Groups, Roles and permissions for **Auth** and **Authz** by CMS Web services and apps

NICA CRIC (SPD)

- Grid description
- Compute and storage settings
- Dynamic configuration
- Report generation, accounting, other grid related apps ??

Datalake

DUNE CRIC

DOMA CRIC

➢ **CRIC** offers a common framework describing generic distributed computing environment with also an advanced functionality to define all necessary VO specific settings

Recent CRIC updates

- Total code refactoring and unification of CRIC engine:
  - ➢ Developed isolated **fwkweb** platform not depended on CRIC (set of applications for generic web framework)
  - ➢ Unified and migrated shared components from `core-cric` into `fwkweb`
  - ➢ Refactoring and unification shared functionality into reusable applications for web services (datatables, syslog, userauth, sslauth, ssoauth, core views, core forms, core processing, frontend templates, metadata, other shared apps ..)
  - ➢ Backport code from `core-cric` into `fwkweb`
  - ➢ Populate fwkweb with new features and components
- Upgrade of dependent 3rd party frameworks/applications for backend and frontend services (Django, Bootstrap, jquery, javascript plugins and libraries, etc):
  - ➢ BS4, BS5 support; Django4
- Implemented CRIC deployment using Docker containers via Docker compose
- Regular updates for CORE-CRIC, ATLAS-CRIC and WLCG-CRIC

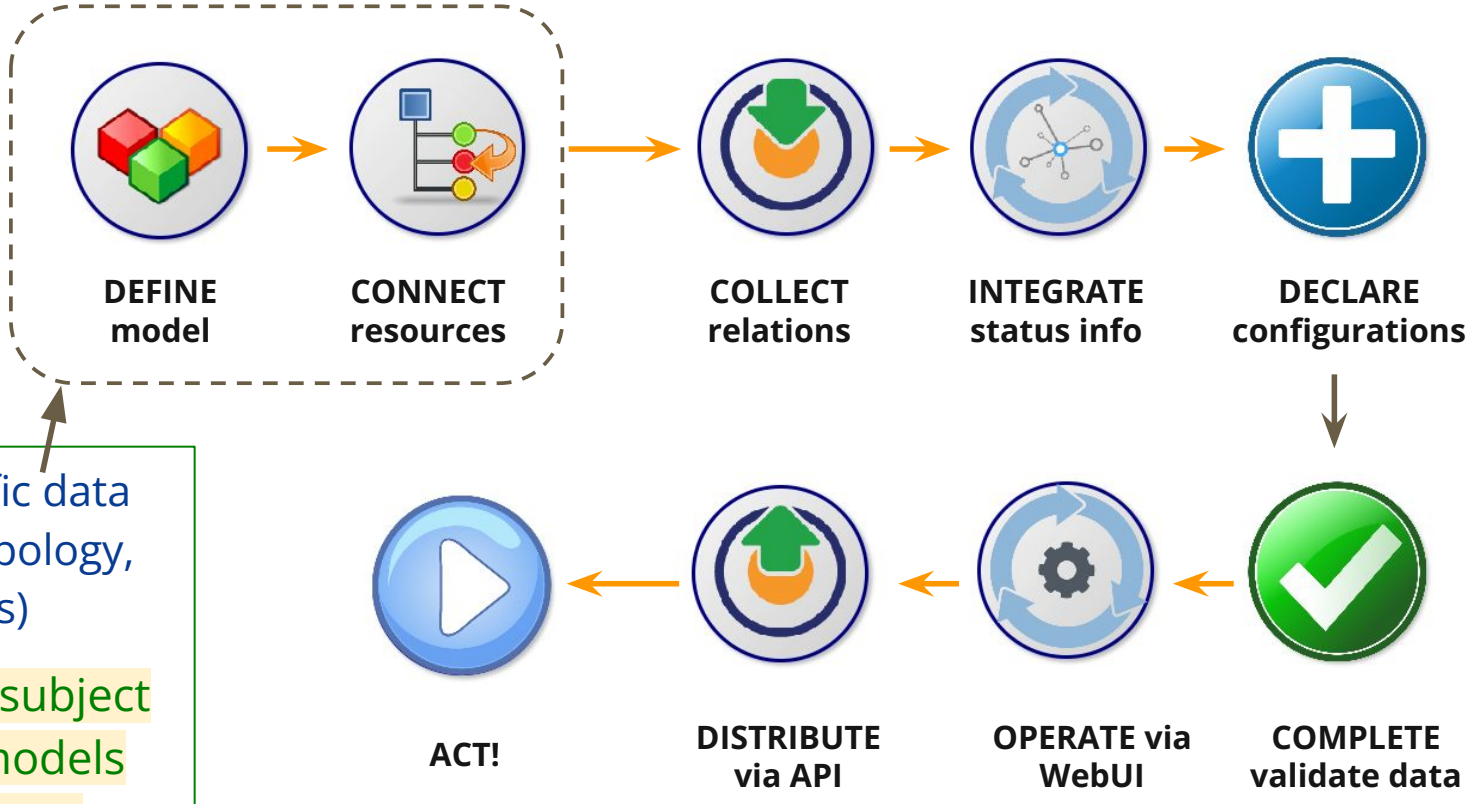
SPD CRIC status

- Up and running in SPD production since 2022
- SPD CRIC relies on ATLAS CRIC implementation; extends and customizes it
- Enabled authentication & authorization via SPD IAM (spd-iam.jinr.ru)
- Upgraded SPD CRIC installation:
  - ➢ refactored and ported CRIC py3 version
  - ➢ affected plugins: ***fwkweb***, *core-cric, atlas-cric, nica-cric*
  - ➢ includes all recent updates and fixes from core and ATLAS
  - ➢ Implemented SPD specific changes in CRIC API (nica-cric)
- Deployed in JINR alma9 cluster using Docker compose containers:
  - ➢ https://spd-cric.jinr.ru
  - ➢ deployment procedure docs: https://git.jinr.ru/spd/spd-dc/cric/spd-cric-operations/
  - ➢ Enabled DDOS protection (fail2ban)
- fwkweb repo: https://git.jinr.ru/spd/spd-dc/cric/fwkweb
- core-cric repo: https://git.jinr.ru/spd/spd-dc/cric/core-cric
- atlas-cric repo: https://git.jinr.ru/spd/spd-dc/cric/atlas-cric
- nica-cric repo: https://git.jinr.ru/spd/spd-dc/cric/nica-cric-py3

Coming back to Data model: CRIC as example

CRIC Information model describes the topology of the Computing models, providing unified description of resources and services used by Experiment applications.

By implementing required Data model for other subject-oriented context we can reuse the concept and build new Configuration system using fwkweb platform



GRID specific data (models, topology, configs)

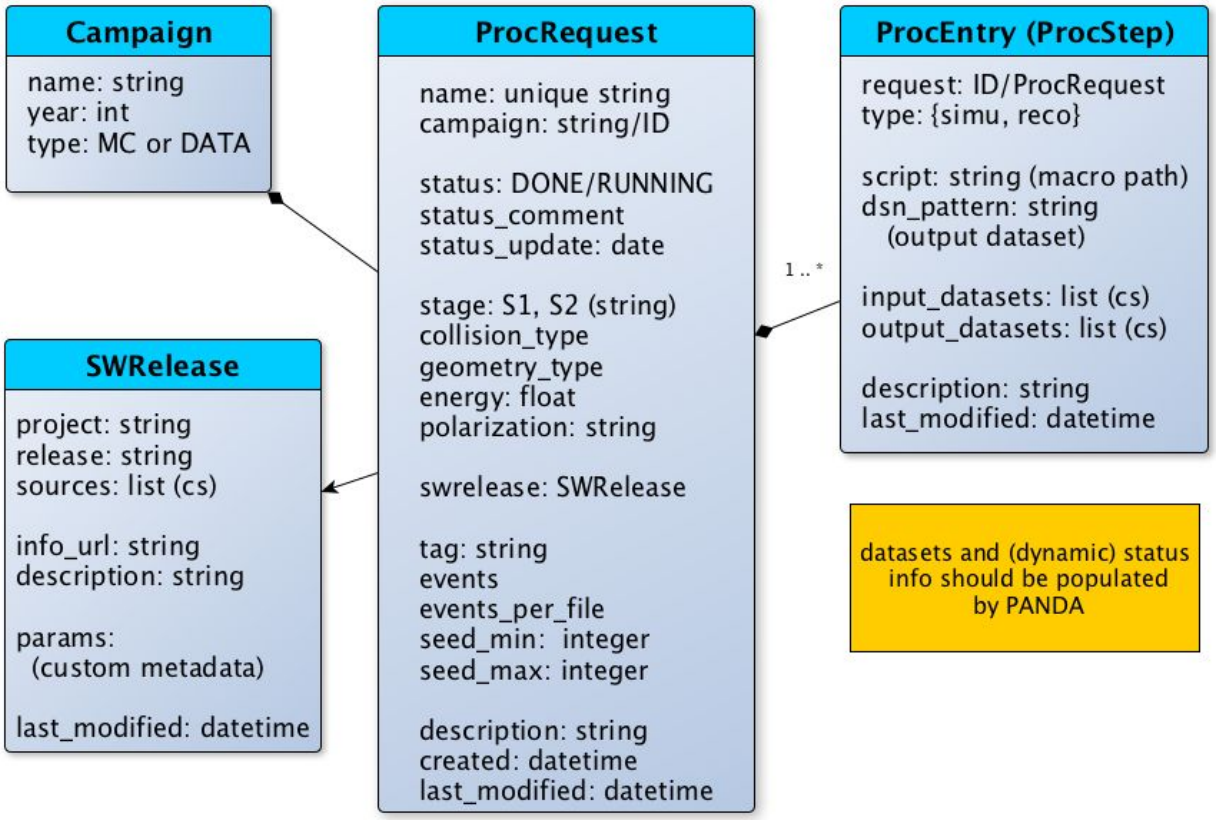
Can be any subject oriented models stored in DB

SPD data processing requests interface (motivation)

- Production of MC data started from March 2025:
  - 2 steps: sim + reco
  - more than 400M+ events produced; 200TB+ data
- Various config parameters required to formalize a production request (e.g. #events, sw releases, seeds, energy, polarization, detector configuration, ..; scripts, input/output datasets, etc..)
- We need to record and keep track of request processing info (status, results, ..)
- Allow config exchange and easy integration with involved SPD Computing components (ProdSys, DDM/Rucio, “Control Panel”, ...)

Requires a DB and a high-level interface system
to facilitate production data processing requests

Information model



- Data Model is focused to store details about Processing requests, the steps of processing as well as keep info about available software versions (SWReleases) with custom metadata associated.
- Dynamic data (status, output ds names) is auto populated from external source(s)
- The model can be easily extended later by adding aux objects (e.g. cache/IDs/refs of PanDA tasks)

Implementation details: inherits fwkweb features

- Dedicated Web application to manage production requests has been developed
  - Plugin based; modular structure; Client-Server architecture
  - Web Services technologies (Client AJAX, REST API, WebUI)

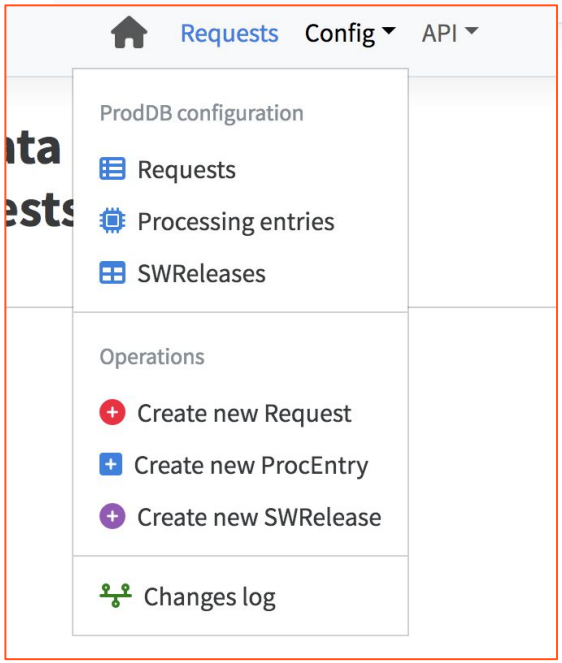
  - Independent DB backends (MySQL, Postgres, Oracle..)
  - Bootstrap framework as HTML/CSS/JS client frontend
  - Apache/WSGI + Python + Django framework as server backend

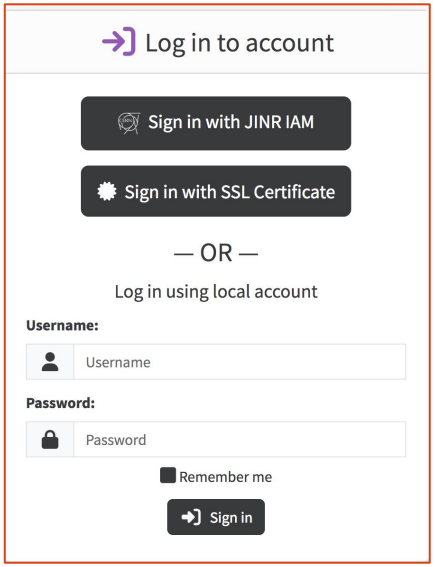
- Employs **fwkweb** platform as a core engine
  (set of web applications for generic framework to build various information component oriented services)

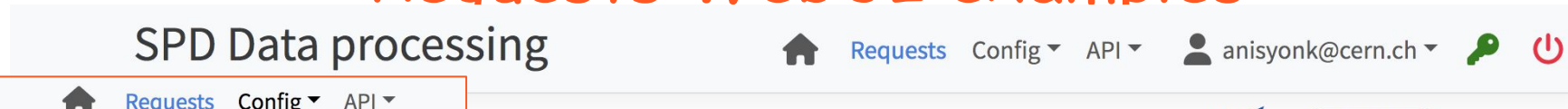
Current Status

- In (pre) production for SPD since Nov/Dec 2025
- Production deployment using Docker via Docker compose
  - spd-requests.jinr.ru
- Deployment docs: https://git.jinr.ru/spd/spd-dc/cric/spd-web-operations
- fwkweb repo: https://git.jinr.ru/spd/spd-dc/cric/fwkweb
- proddb repo: https://git.jinr.ru/spd/db/metadb/proddb

Requests WebUI examples

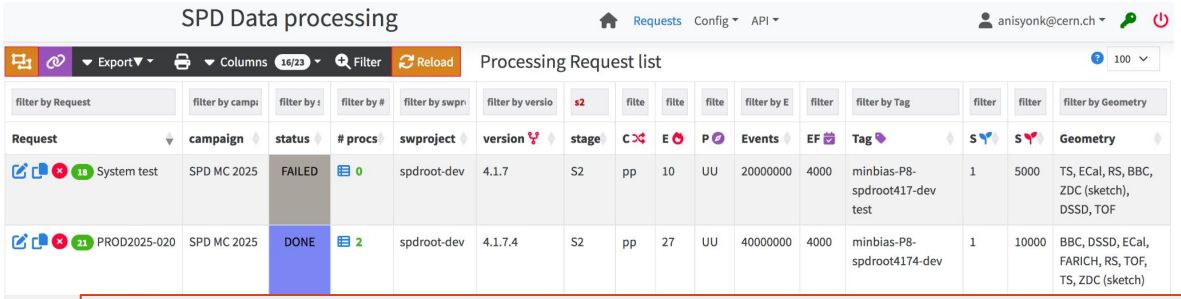


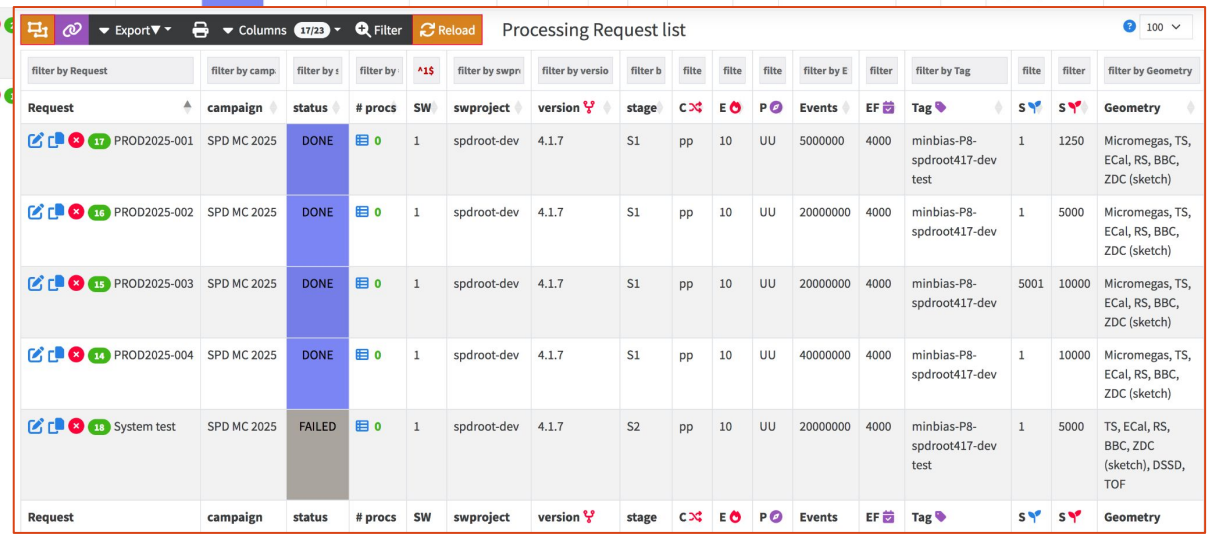




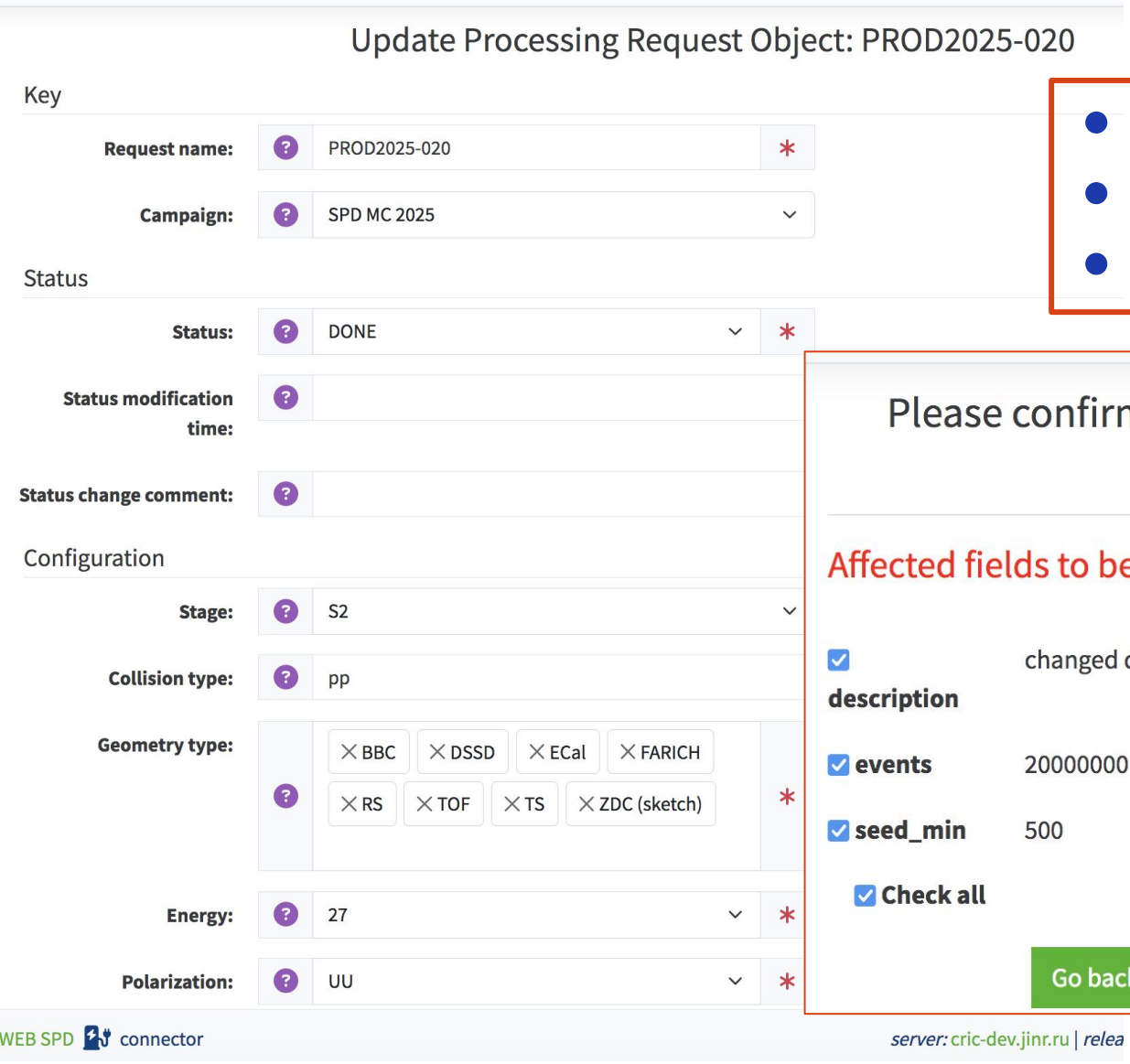
SPD Data processing
Requests interface

- Interactive filtering, live sorting
- Selection by related SWRelease, RequestID

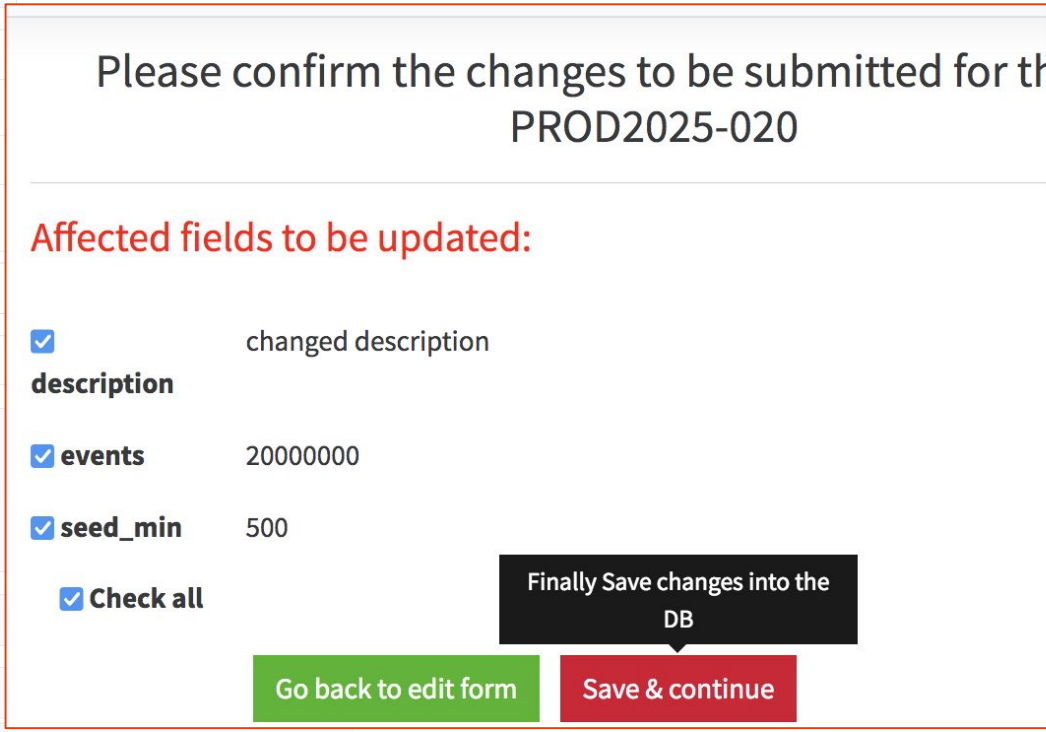




Update forms examples



- Built-in data validation
- Changes confirmation
- Clone object functionality



daqweb family
(DAQ Operations and monitoring)

Web applications: daqweb framework

- Independent project for DAQ applications: Configuration, Monitoring and Operations
- Developed in parallel within CRIC era sharing same concepts, methodology and somewhere completely reuse cloned modules from CRIC and vise versa
- daqweb follows same design concept of splitting project into set of standalone applications which can be shared and extended later.
- Similar system with CRIC in terms of engine implementation (Django-based web service, REST API, plugin-based, modular structures)
- Regular technology transfers between daqweb and CRIC projects, regular sync of shared code components and updates.
- daqweb introduces new features specific for monitoring (interactive plots, data visualization, templatags widgets, hardware controls, etc..)
- Cross experiments: in production for **CMD-3 Collaboration**; using by BINP **MRT X-tomography** stations; worked dur run-1 in **Muon G-2 at Fermilab**
- Currently is being refactored to rely on **fwkweb** components

Brief introduction into subject area: Role of monitoring tools in DAQ

Slow Control and monitoring system is a vital part of any HEP experiment

- Monitor the status of DAQ and DAQ hardware
- Monitor physical and environmental conditions
- Control the quality of data taken
- Control and operate hardware equipments

- Guarantee safety and correct functioning of whole system

Basic sources of monitoring data

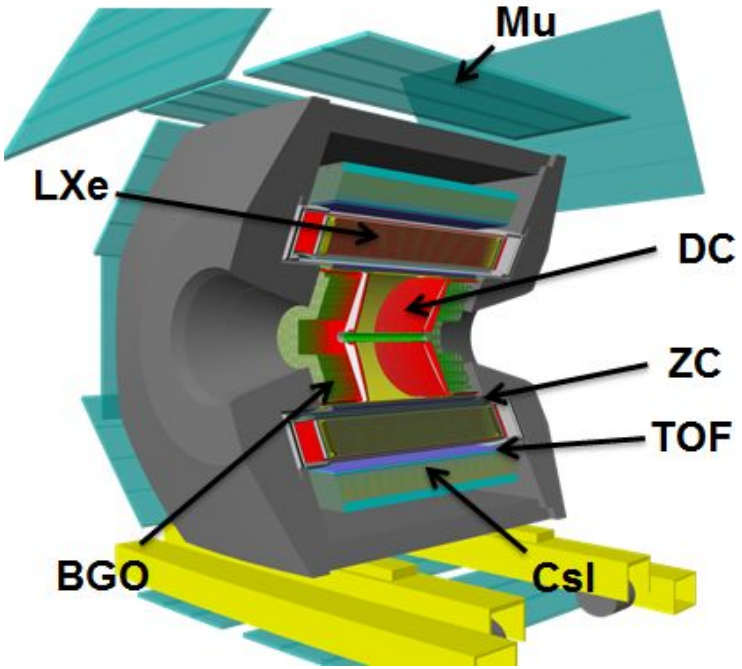
During the operations DAQ and related systems produce a lot of information for experts and people on shift that need to be monitored and taken into account

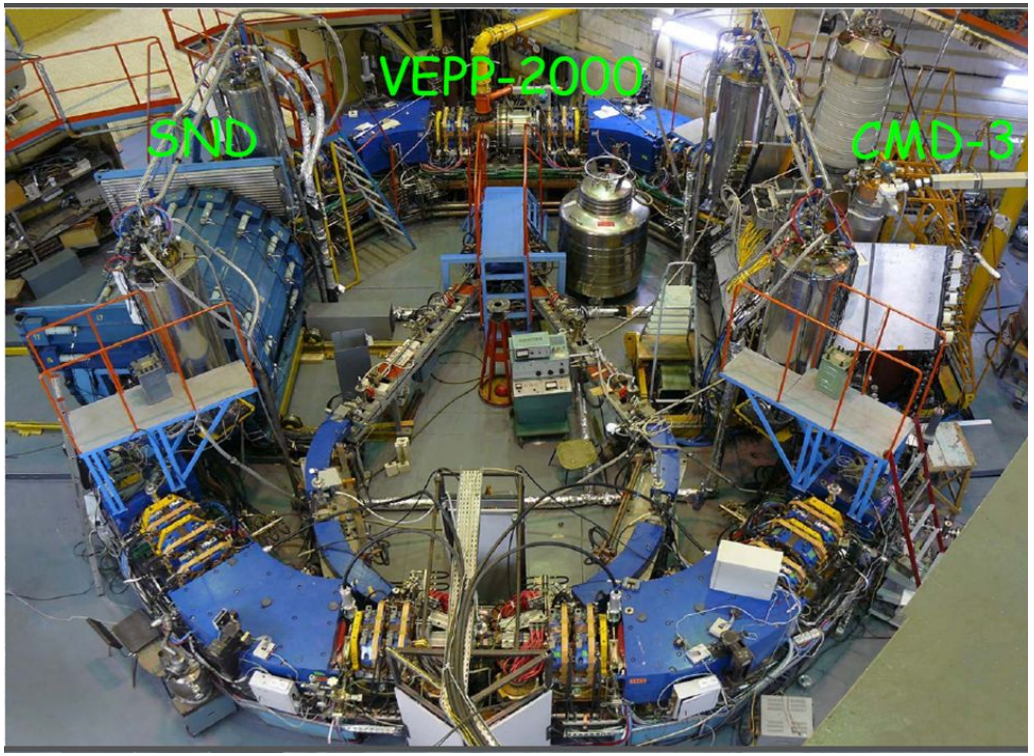
Slow control	Offline Reconstruction	Online DAQ
Centralized Slow control hardware sensors	Online Data quality metrics	Direct read-out of front-end electronics (crates, subsystems)
Subsystems monitoring channels	Slow Control software sensors	DAQ status metrics; Run Log
Archived Slow Control data	Offline Data quality metrics	Nearline data processing

CMD-3 Experiment

daqweb **system discussed in the talk was developed for CMD-3 detector**

Typical small-to-medium scale HEP experiment

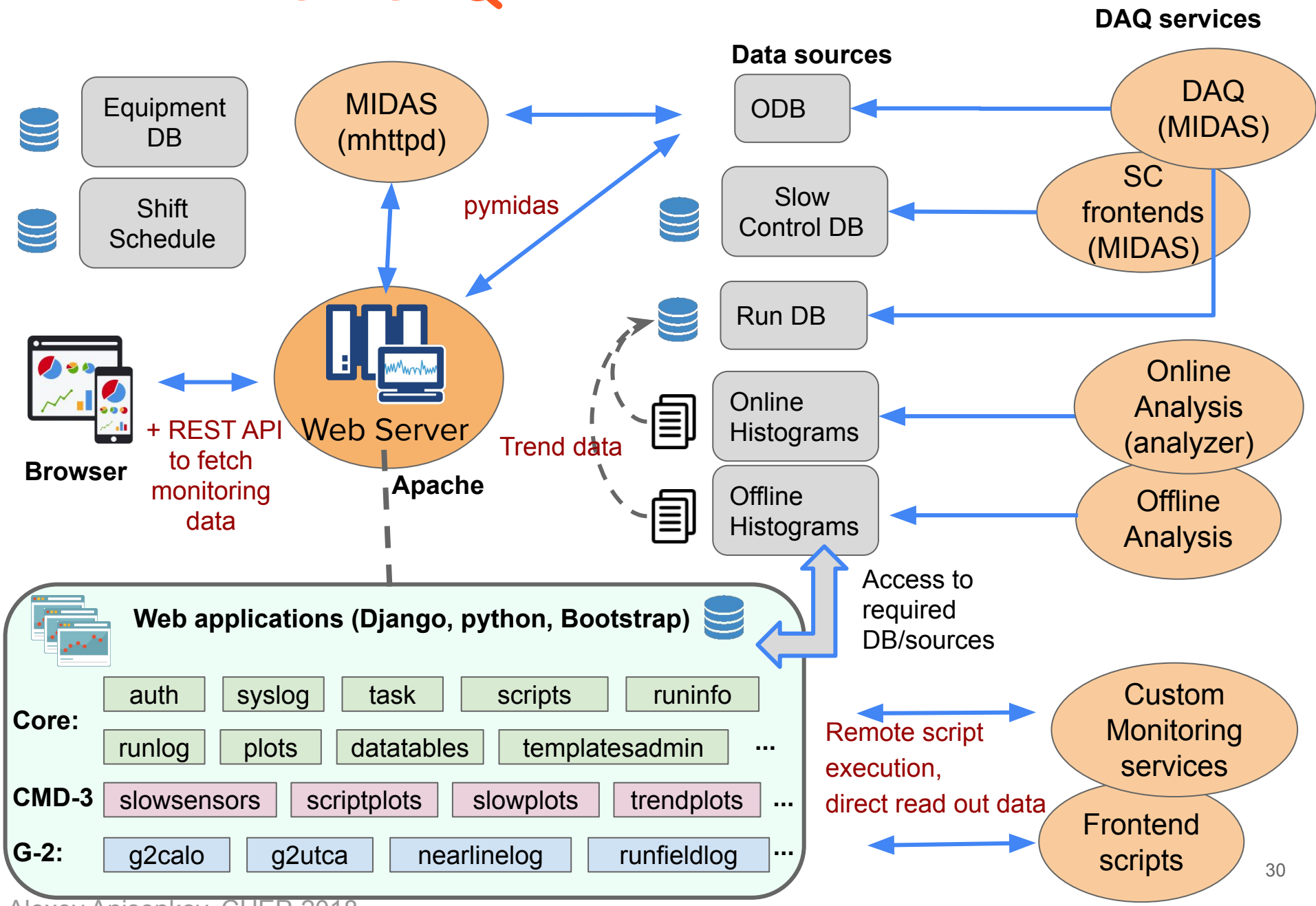




- e+ e- collider VEPP-2000 at BINP (Novosibirsk)
- 7 detector's subsystems + cryo, gases, HV, LV
- ~ O(1000) environmental sensors
- ~ O(100) monitoring histos, data quality plots

- 60 authors
- 10k event size, 1kHz FLT rate

CMD3 DAQ: Architecture overview



Graphical component to draw plots

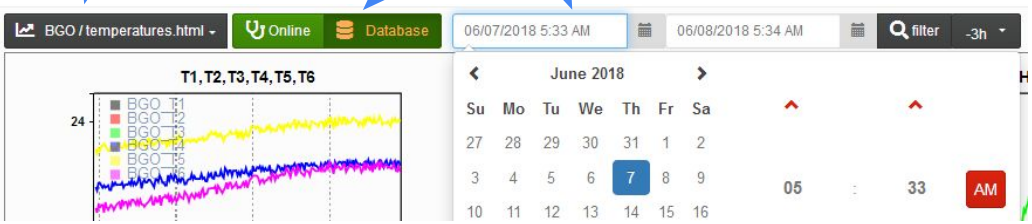
Own implementation of low-level plot.js widget based on D3.js

- Fully interactive, dynamic data visualization
- Data loading via REST JSON API
- Implemented as standalone JQuery plugin
- Draw several graphs on same pad within canvas
- Common X-axis slider for all plots on a page
- Predefined time windows
- And more..

Interactive plots: some features

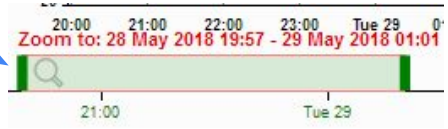
Predefined plot presets

The same interface for real-time and historical data

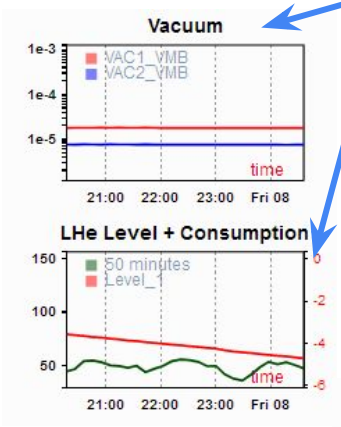


Automatic refresh for real time data

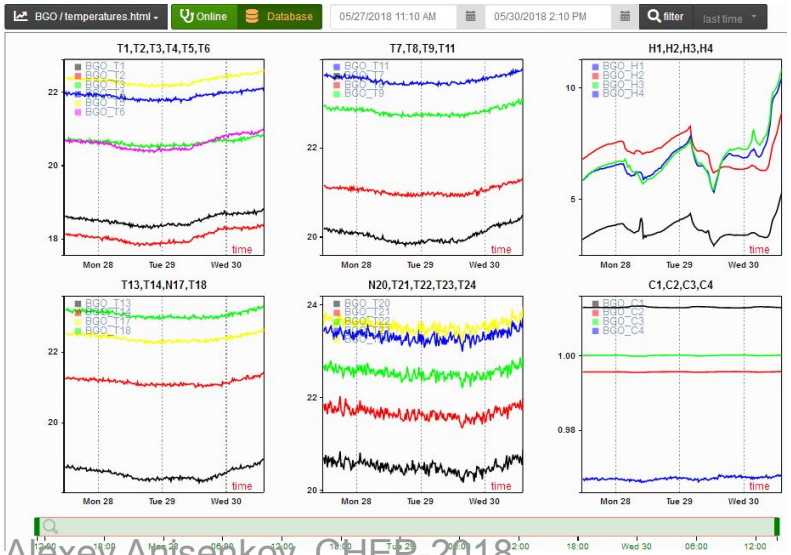
Ability to zoom in/out for x,y axis to get more detailed picture



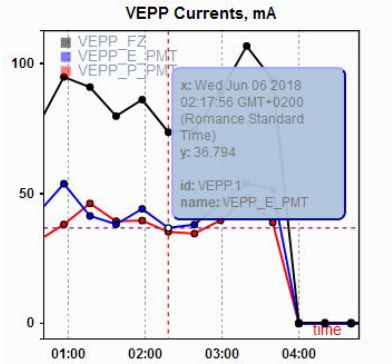
Log scale, 2 y-axis on same pad, custom data transformation (deriv)



Slow plots (time as x-axis)



Automatic zoom and switch from lines to points level depending on requested x-time window;
Point details pop-up window

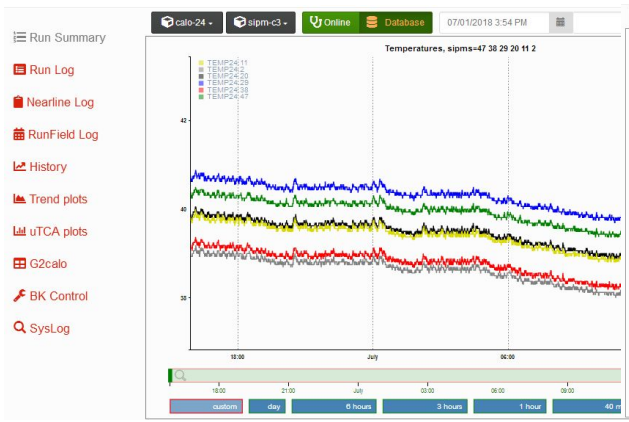


Graphical component: shared implementation

Given plots application is used as a base engine for following components:

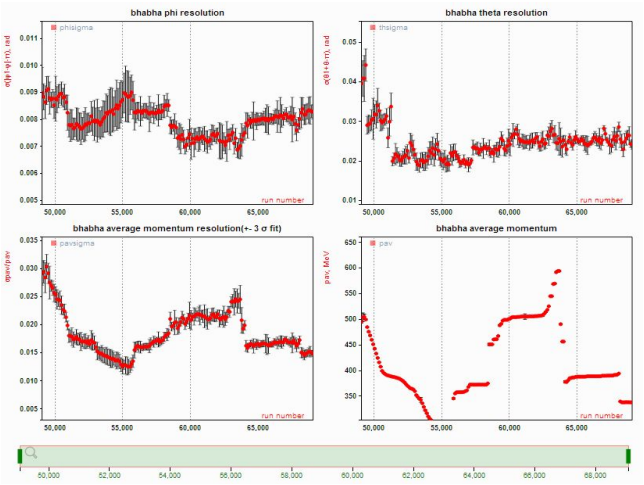
- Central Slow control data visualization (**slowplots**)
- Online and Nearline analysis data visualisation - run by run trending (**trendplots**)
- Custom data monitoring
  - Real-time read-out from frontend electronic (e.g. temperatures of SiPM calorimeters at G-2 - **g2calo**)
  - Draw monitoring data from custom db/source (e.g. monitoring of microTCA crate temperatures/params at G-2 - **g2utca** application)

G2calo plots



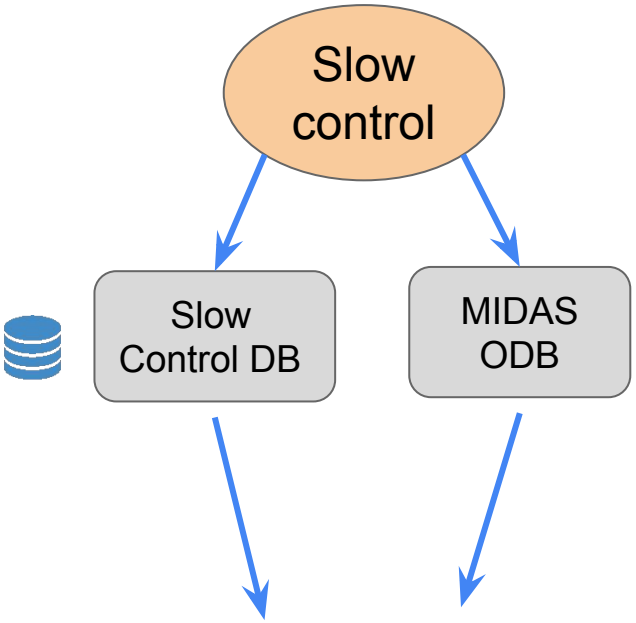
G2utca plots

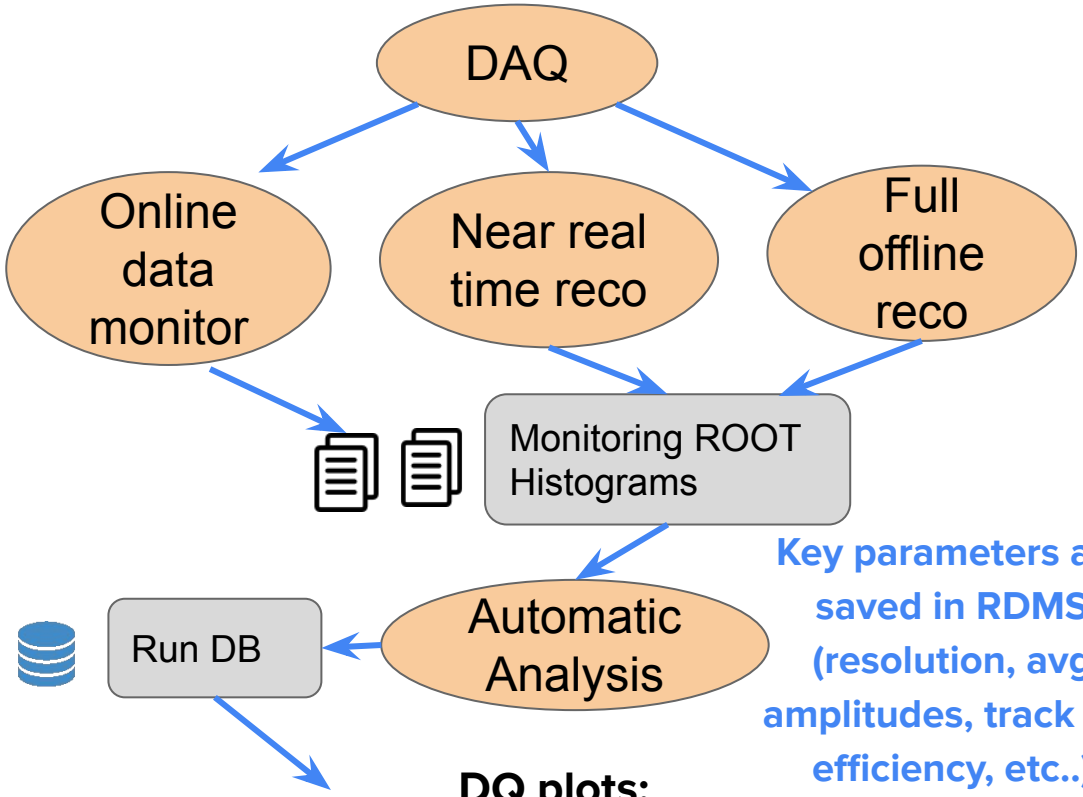
CMD-3 trend plots



Data quality plots (trend plots)

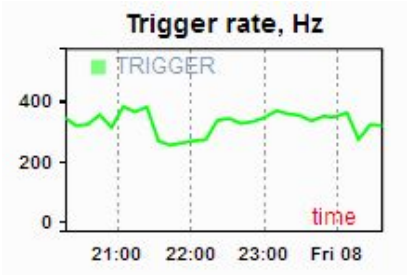
Different data flow to generate data quality metrics (online, nearline, offline)





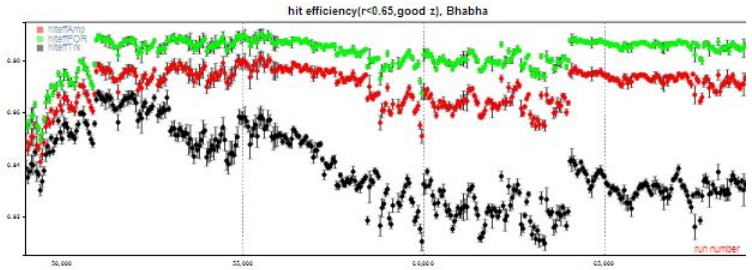
Key parameters are saved in RDMS (resolution, avg amplitudes, track rec efficiency, etc..)

Slow control plots:



Values vs time

DQ plots:



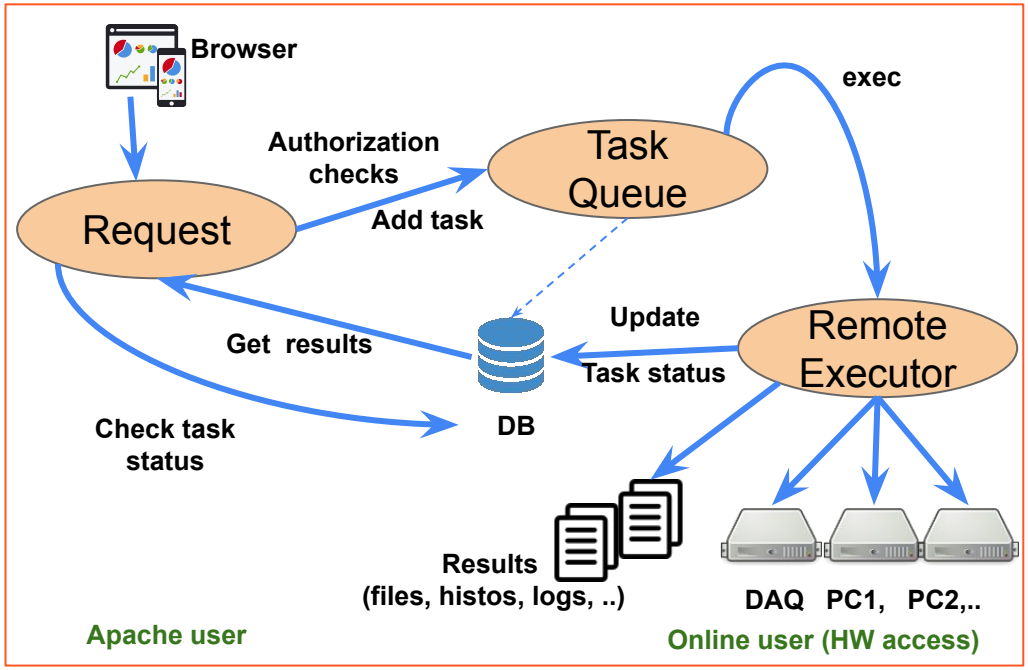
Values vs run #

Remote script execution

The system is able to execute custom scripts from the web page, run them real-time at required DAQ machines, and report exit code/stderr/stdout back

Base scripts component:

- Use distributed task queue Celery + MySQL/RabbitMQ as message broker
- Register within the system corresponding Task and track its status in WebUI
- Use **template tags** approach to customize how data should be reported back to web
- Support for locking (multiple launch protection) + appropriate authorization checks



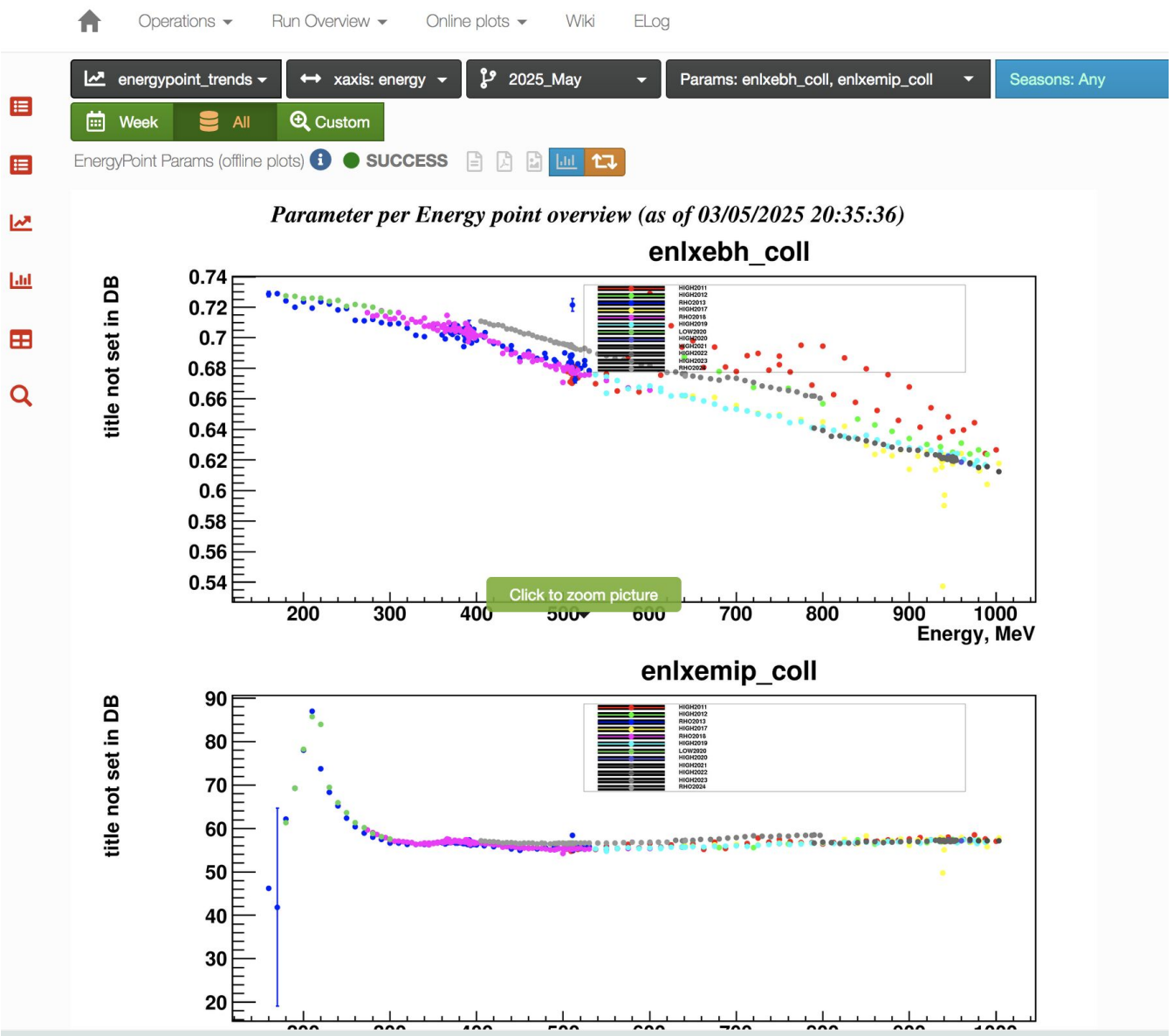
Typical use-cases and applications:

- Interactive hardware control (e.g. prepare boards for data taking, **runscripts** at CMD-3)
- To generate histograms/plots server-side with complicated analysis or involved several data sources using ROOT/JSROOT
  (e.g. **scriptplots, offlineplots** at CMD-3, **trendplot** at G-2)

ROOT Data Analysis Framework

daqweb: Example of plots in Offline apps

- **Based on Remote script execution**
- **Dynamic input configuration (what params should be drawn, what axis, reprocessing campaign, etc..)**
- **Draw engine by ROOT**
- **Static images, But can be visualized as well from ROOT files client side**



Runlog table view/operator helper (classic application)

Provides list of collected runs during shift with primary information exposed

- Interactive view to browse Run log table operated by MIDAS
- Ability to update Run details if need

Live filtering, customize columns, resolve runs by given shift/date

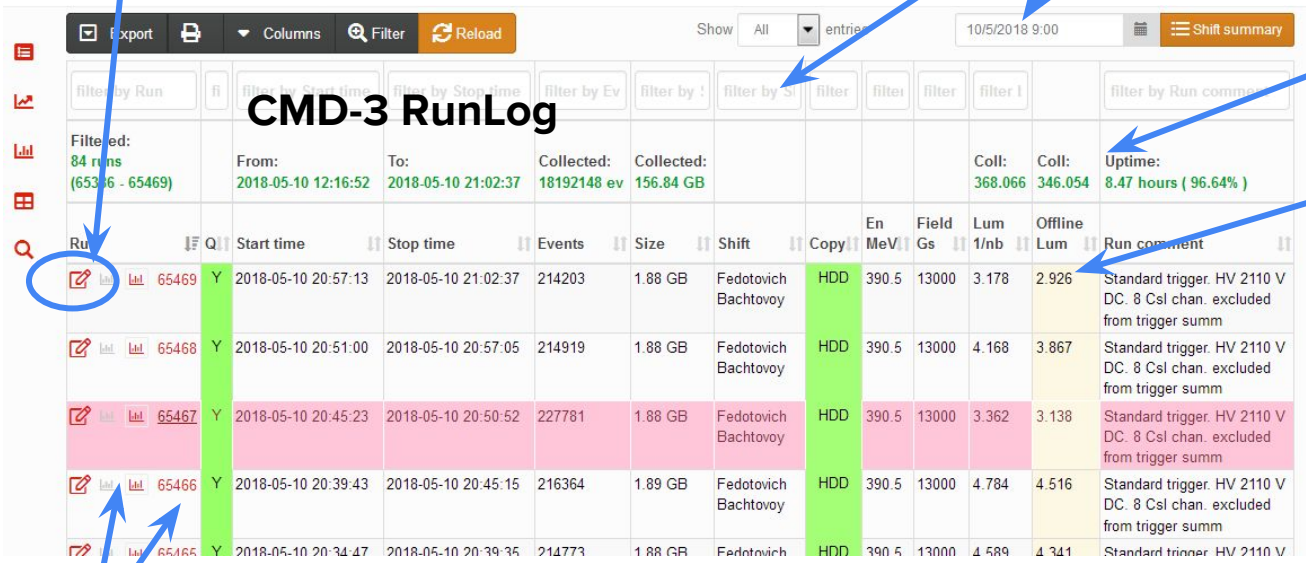
Provide shift overview in numbers

Complement Run details with parameters produced by Offline Analysis

Highlight bad Runs that require attention by operator

Links to online histograms and Run passport page

Same tableview as in CRIC



Run Field Log at G-2

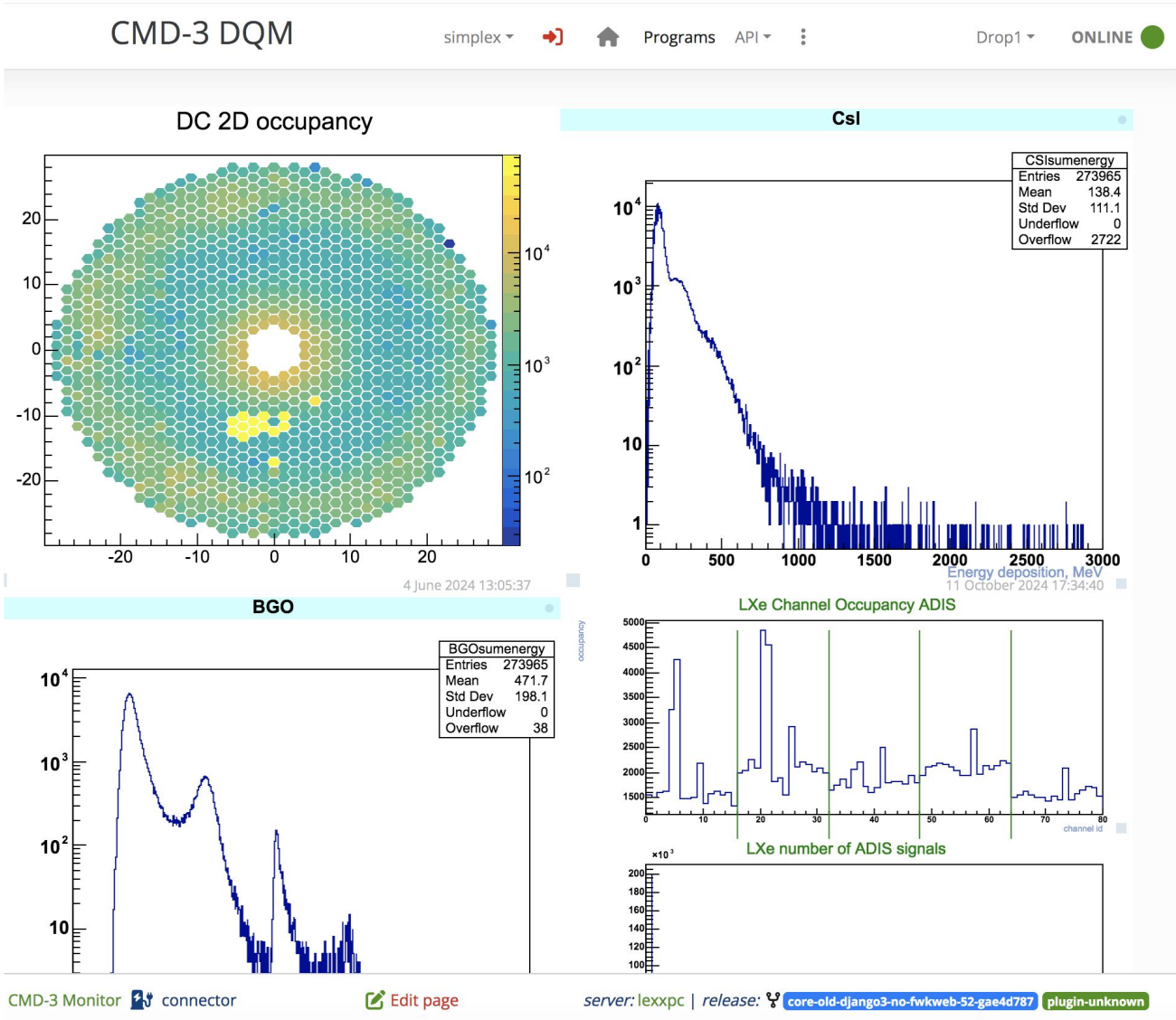
Filtered: 50 runs (5267 - 5317) | From: 2018-06-26 07:16:44 | To: 2018-06-30 10:47:15 | Uptime: 90.08 hours (90.53%)

Run	Start time	Stop time	Quality	Field (kHz)	PSFB Curr. (mA)	Trolley pos	Trolley mode	Flux	PP	PS FB	PS Active	SCC	Trolley	Galil	FP	Run comment
5317	2018-06-30 09:03:32	2018-06-30 10:47:15	Y	48.48	0	175	Idle	✓	✗	✓	✗	✓	✗	✗	✗	Magnet ramping down
5316	2018-06-30 08:26:27	2018-06-30 09:01:43	Y	48.52	0	175	Idle	✓	✗	✓	✗	✓	✗	✗	✓	Field run after magnet recovery
5315	2018-06-30 06:24:13	2018-06-30 08:26:14	Y	52.03	13.5	175	Idle	✓	✗	✓	✓	✓	✗	✗	✓	Field run after magnet recovery

Fwkweb: Event-based async DQM

fwkweb + daqweb engine (py3)

- **Ability to subscribe for required channels and receive immediately updates once produced by publisher**
- **DQM histos collected real time by Online Analyzer using ROOT**
- **publisher can produce whatever aggregated statistics or images**
- **Fully interactive images, Client side visualization**



Fwkweb platform usage examples

fwkweb

daqweb family

DAQ

- RunDB
- OfflineRuns
- SlowControl
- DQM
- Monitoring
- OnlinePlots
- NearlinePlots
- ConfigDB
- ….

Online, Nearline, SlowControl, Offline applications

SPD production requests

SPD NICA

CRIC family

CRIC

Computing Resource Information Catalog

SPD CRIC

NICA

ATLAS CRIC

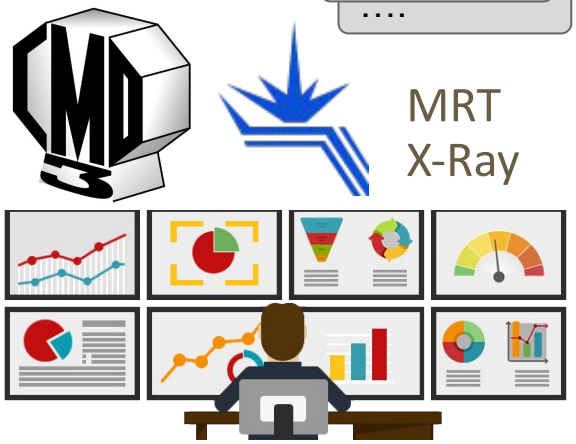
ATLAS EXPERIMENT

WLCG CRIC

CERN

MRT

X-Ray



fwkweb: DAQ, Control, Monitoring, information component oriented applications

Conclusion

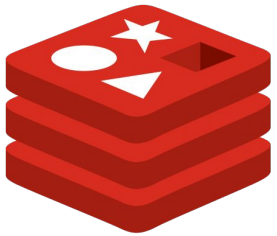
- **W**eb technologies can be effectively used to build functional, handy and attractive applications in various fields of Computing activity
- **CRIC** offers an Information framework describing Distributed computing environment with advanced functionality to define all necessary Experiment-specific configurations
- **SPD CRIC** is up and running as part of Grid middleware stack
- **fwkweb** platform can be helpful to build custom configuration system for specific data models beyond GRID subject area:
  - First example: SPD data processing requests interface (running)
  - SPD production system control panel (ongoing implementation)
  - SPD Collaboration DB (ongoing implementation)
- **daqweb** platform is another example of Web-apps in DAQ and monit.
  - Actively used by CMD-3 Collaboration in Online, Nearline and Offline
  - Provides full access to various monitoring data, lightweight visualization, as well as possibility to configure hardware

Thank you for your attention!

- https://spd-cric.jinr.ru (SPD CRIC)
- https://spd-requests.jinr.ru (SPD data processing requests)

Fwkweb updates: async wsgi support (2024)

- Web sockets support for Django applications
- Real-time (bidirectional, low latency, event-based) communication between a browser and server(s)
- Lightweight production deployment using python native uWSGI server with async capability by Socket.IO + gevent (event loop) + Redis cache
- Possible applications: real time Data Quality Monitor, Event-based data visualization (statistics)



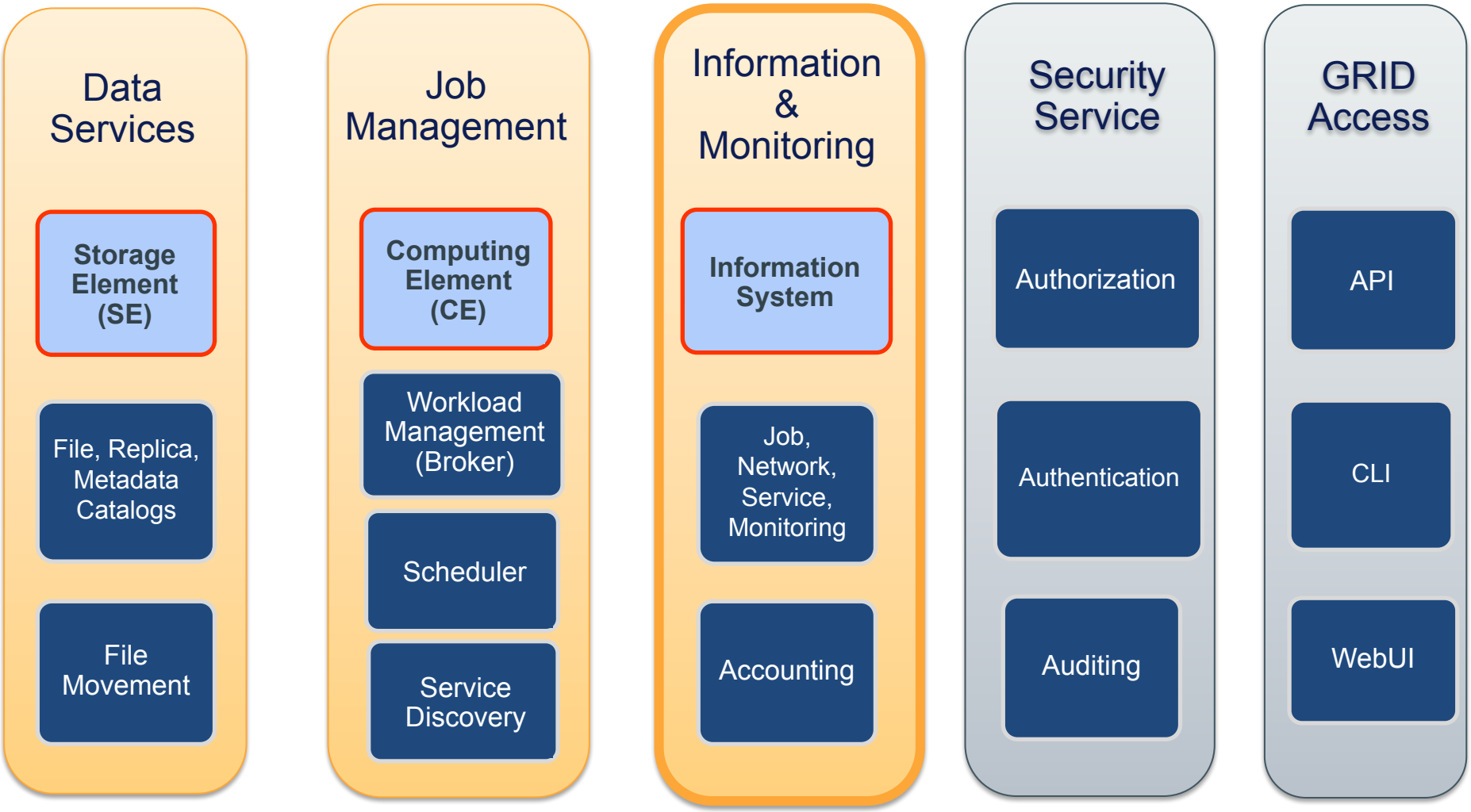
CRIC features as the infosys middleware for VOs

- Helps to easily integrate new Computing technologies which have not yet appeared in WLCG as the services or can not be part of WLCG in general, ATLAS examples:
  - newer type of SE based on ObjectStore technology
  - Federated Access to storage (FAX redirectors, direct access to remote files from Worker Nodes)
  - Description of opportunistic/volunteer resources

- Helps to minimize side effects for end-user applications of various internal migrations/changes/tests/evolution of Distributed Computing components/infrastructure:
  - Consolidation of protocols description that should be applied only for few sites, unification of resources, migration to HTCondor
  - Keeps data export in several format for backward compatibility

- Masks incompatible updates in external data providers, implement missing functionality/overwrite/fulfill data:
  - e.g. fix wrongly published number of cores, core-power
  - remove direct dependency to ext sources (obsolete data providers)

Basic Components of Grid Middleware



Implementation feature: Django template tag as widget

Special template tag encapsulates all complicated logic and allows easy configuration of plots by users within WebUI

We use Django tags to create “widgets”

- Pages can be edited directly (thanks to **templatesadmin** app implemented)
- **slowplot** template tag specifies plot configuration (sensors, pads, colors, ranges, axis settings, transformations, auto zoom, etc..)

Edit Template: slowplots/presets/Run_Overview/Environment.ht

```
{% extends "slowplots/_base_preset.html" %}
{% load slowplot %}
{% load fields %}
{% block title %}Environment{% endblock %}

{% block main_content %}{{block.super}}

{% slowplot name='env' nx=2 ny=2 source="{{source}}"%}

source_url = {{declare.sources|get:source}}
refresh_cid = {{declare.refresh_cid}}

pad.0.title = Temperature, Celsius
pad.0.sensor.0.name = ENV.0

pad.1.title = Pressure, mm
pad.1.sensor.0.name = ENV.1

pad.2.title = Humidity, %
pad.2.sensor.0.name = ENV.2

{% endslowplot %}

{% load owner %}
{# keep the owner tag below to control who will be able to modify
this template #}
{% owner user='anisyonk' group='DAQ,ALL' %}

{% endblock %}
```

Backup file before saving? ☐ SAVE