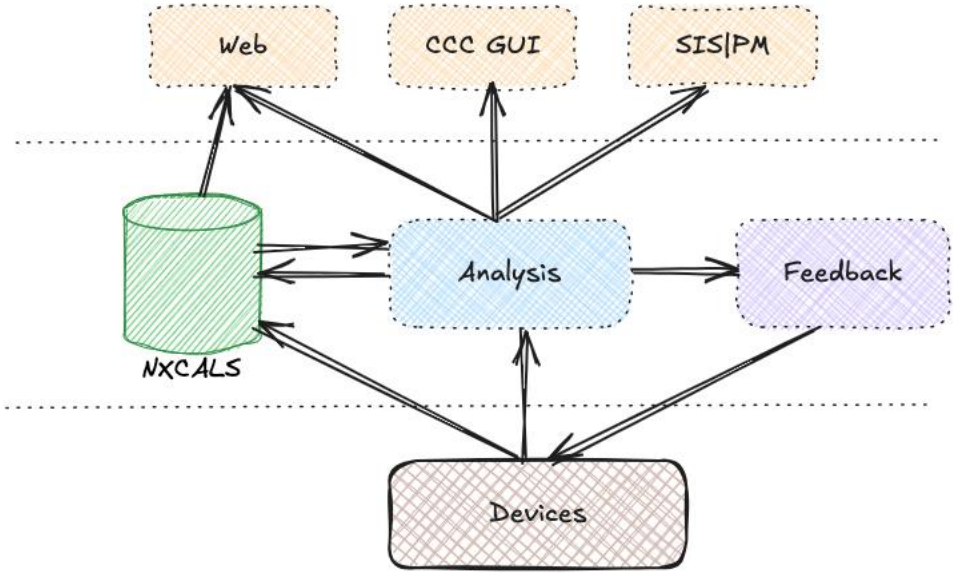
Data and processing resources for analysis

Marcin Sobieszek BE/CSS
EPA WP9

with input from
G. Iadarola, G. Sterbini, R. De Maria, F. Asvesta, A. Lasheen, M. Berges, JC. Garnier, V. Kain, A. Huschauer, M. Schenk, M. Hostettler, A. Calia, G. Trad, V. Baggiolini, C. Roderick and others

Overview



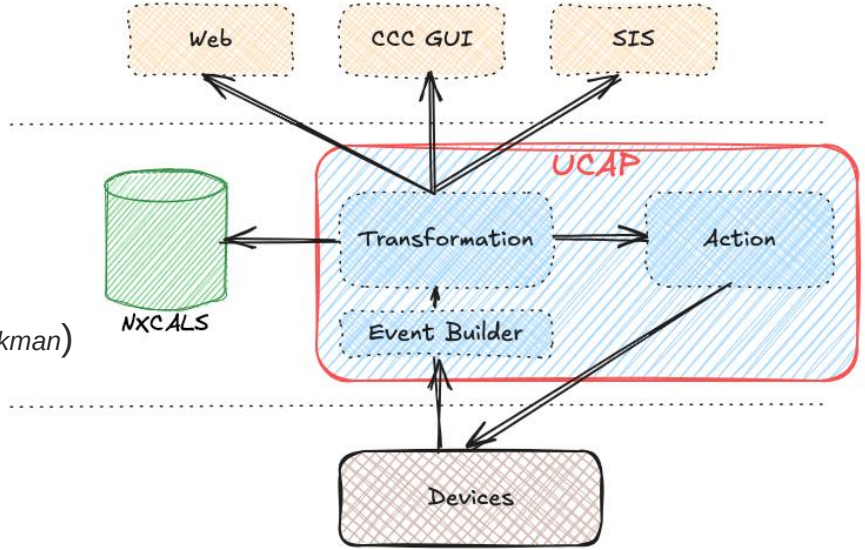
Online Data Processing

UCAP – the primary choice for online data processing

Common pattern: Data aggregation → Process → Publish + Store [Notify]

Many different use cases:

- Processing & concentration of operational signals
- Wire Scanner analysis
- Diamond BLMs
- Automatic Fault Recording (*see talk by A. Asko*)
- SISv2 complex logic
- Automated multi-bunch tomography (*see. talk by A. Beeckman*)
- etc...



UCAP overview

10 bare-metal machines (***cs-ccr-[1:10]***)

2 machines with GPU (***cs-513-ml002, cs-ccr-ml004***)

Node count: **203 nodes** : 95 PRO, 102 TEST, 6 TESTBED

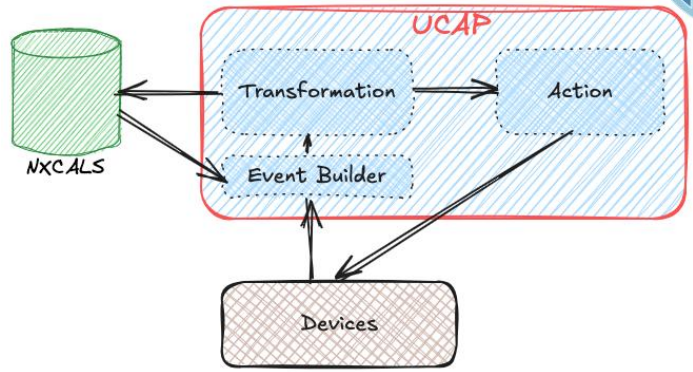
UCAP extensions and improvements

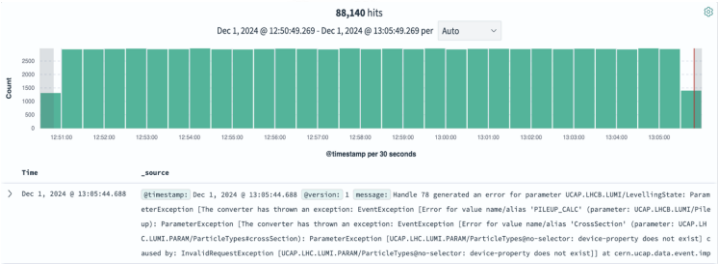
Recent developments driven by EPA & community

- Combine online and offline data → integration with NXCALS
- Simplifying debugging → logs are now stored in OpenSearch
  - Still room for improvement for Python (debugging from IDE)
- LSA Trim, AFT, Email Actors, SET support, and many more.

Planned improvements

- Automatic device declaration in CCS
  - No more manual declarations in the CCDE web interface
- *ucap-persistence* to keep Converters state when UCAP node is rebooted
- Support for event recording & replaying
- Streamline configuration with automated JSON generation





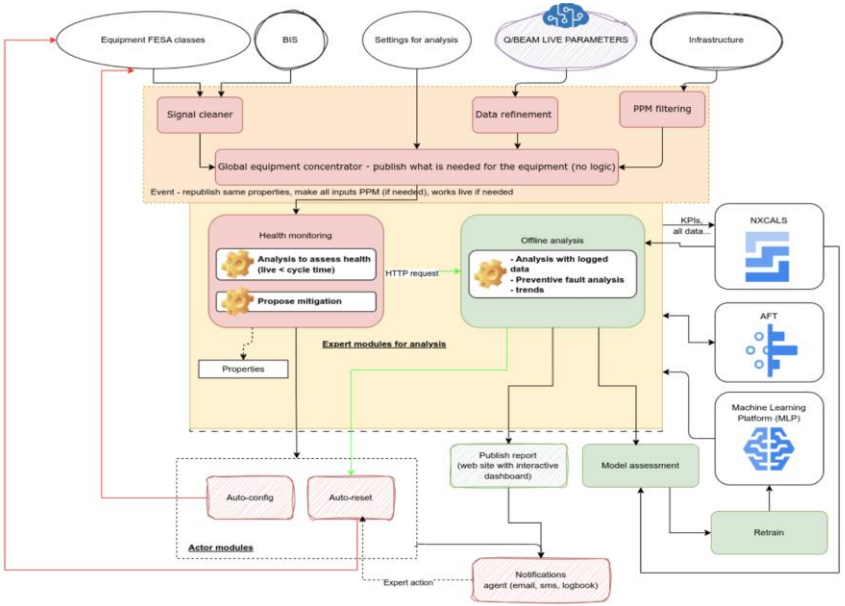
Limitations:

- UCAP is not a virtual device framework, i.e. unsuitable to implement e.g. Orchestration server → We need something else
- In UCAP all data is expressed in JAPC-style, however, operating on domain-specific types would be more convenient

New Data Processing Requirements from EPA

EPA WP8 – towards equipment automation aka “smart and agile equipment”

A framework that allows equipment experts to design automation for their machinery



Courtesy to F. Velotti and K. Papastergiou (JAP23)

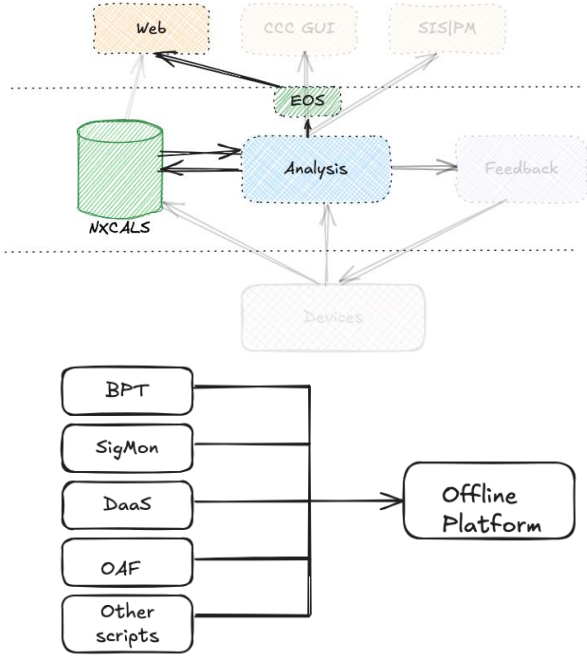
=> A more powerful and flexible analysis framework is needed

Offline Data Processing

Offline Data Processing Overview

Common pattern: Query data from NXCALS → process → publish + store + visualize

- Several different implementations exist with similar functionality
  - **O**ffline **A**nalysis **F**ramework – regular analysis of instrument data for tracking performance, aging, etc. (SY-BI)
  - **B**eam **P**erformance **T**racking - operation performance of the different accelerators (BE-OP, BE-ABP)
  - **Sig**nal **Mon**itoring – hardware commissioning, powering tests, circuit performance, quench analysis (TE-MPE)
  - **D**ata **A**nalysis **a**s a **S**ervice – water consumption, Cryo instrumentation and operation (BE-ICS, TE-CRG)
  - Several other systems from ABT, STI, TE-VSC
- Conceptually similar but with different implementation
  - mainly written in Python, different frameworks chosen
  - often operate on ad-hoc infrastructure that might not always be reliable enough
- Endorsed by the CTTB => Converge and provide a common solution



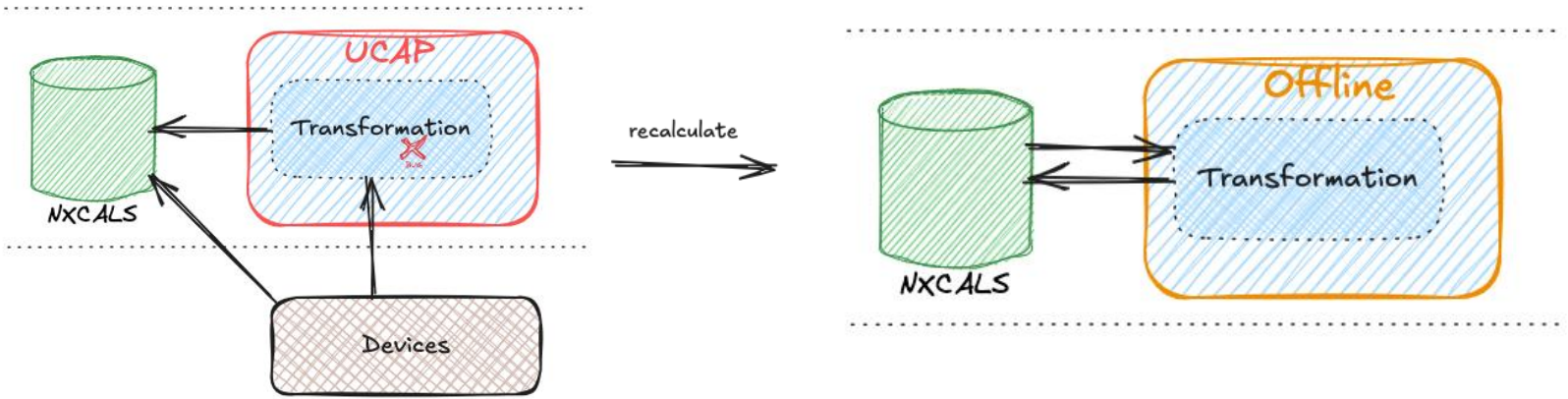
Vision/ideas for a unified offline solution

- **Ideally all data that needs to be analysed is stored in NXCALS**
  - Data from devices ✓ (Data can be logged on-demand, e.g. only during an MD)
  - LSA settings ✓ (no need to use pjLSA)
  - Optics and other information ➔ to be addressed
- **Analysis scripts**
  - Are written in any supported language (e.g., Python, Java)
  - Triggered by a fixed schedule (cron) or an external event (e.g. End-of-Fill)
  - Results can be stored in NXCALS, published (CMW, email etc)
- **You (users) take care of:**
  - Developing the analysis scripts and defining its configuration (e.g., trigger, result publication etc.)
- **We take care of all the rest**
  - Infrastructure, monitoring, upgrades
  - APIs, integration between services, deployments, triggers ...

Bringing offline & online together

Online and offline – why and how to merge?

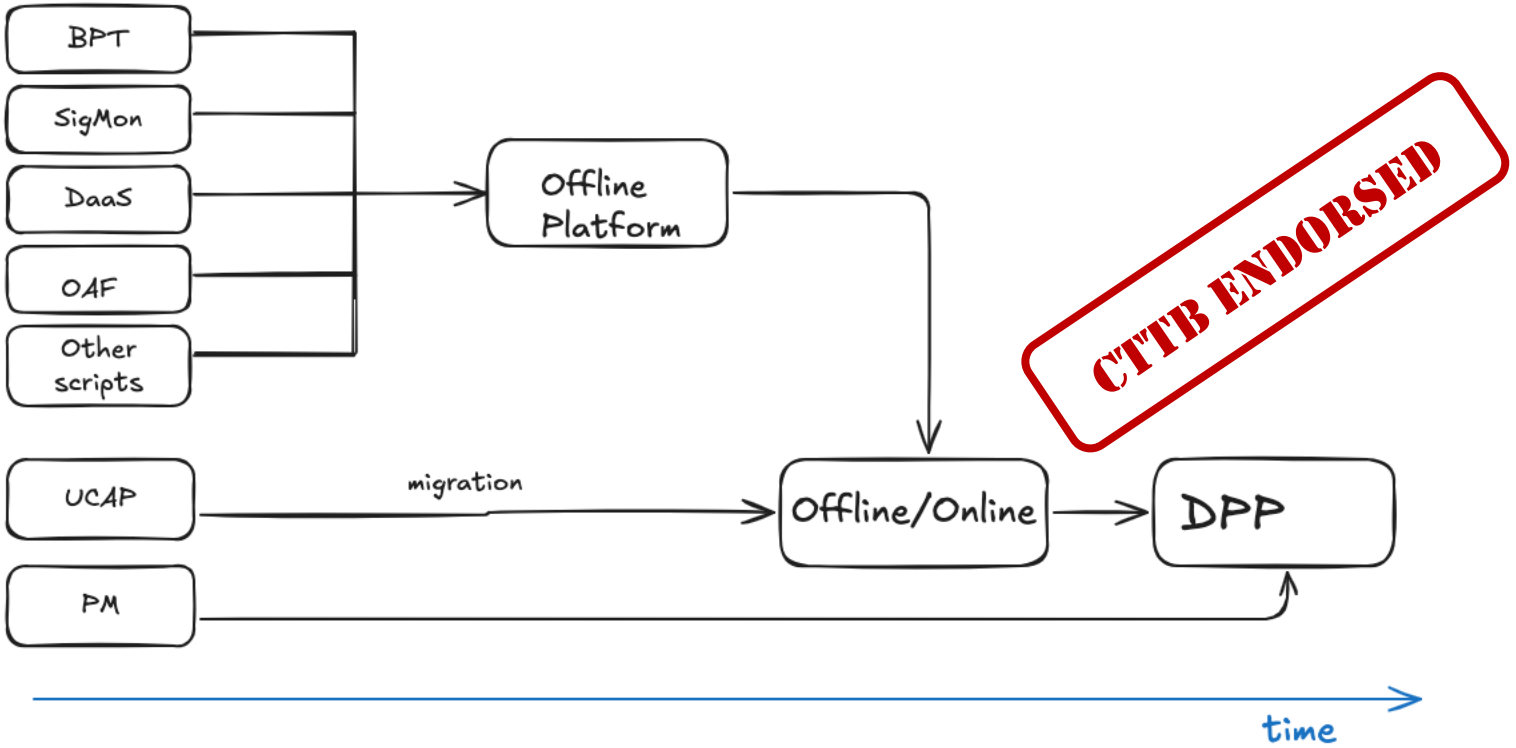
- Having distinct solutions for online and offline is inefficient
- Having the same (e.g., SWAN?) algorithm used for online and offline could be very useful



- **=> These two approaches are complementary and should be combined.**

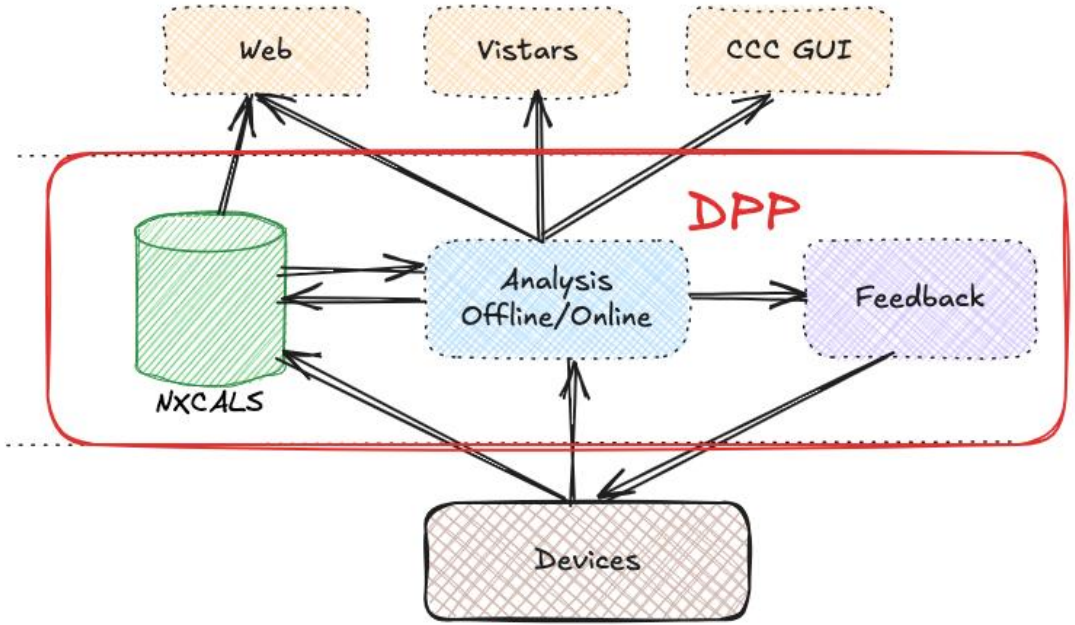
Challenges to achieve this: Finding a common way of expressing an algorithm for online and offline == common tools, data structures and APIs

Bringing online & offline together



Data Processing Platform

One tool for Online and Offline Analysis adopting the UCAP philosophy: focus on solving your problems and performing your analysis – we'll take care of the rest



Miscellaneous

- NXCALS: More sophisticated *logging-on-demand* needed for MD
  - Support for predefined time windows (e.g., for the next 8 hours)
  - Nice to have: conditional logging triggered by external events (e.g., when there's a beam)
- NXCALS: Data post-processing, down sampling, repartitioning
- NXCALS: Simplified API, UTC → local time, pandas for seamless Analysis
  - Help people use Spark efficiently
- NXCALS: Even more data into NXCALS (Optics + Settings + Meas) for even more seamless Analysis
- UCAP, NXCALS: Even more support for Python
  - "*if Python was not supported we wouldn't use UCAP at all*"
  - *"Do we actually need Java in Client Facing API?"*
- MISC: Controls orchestration service strategy - clarify needs and roles between DPP and Sequencer 2.0 (WP5)
- MISC/DPP: *EventBuilder* as a specialized Python library or a service
- MISC: API for cross-accelerator correlation (e.g. to follow an LHC bunch back to the PS cycle that produced it)

Plans for 2025

UCAP

- Implementation of *ucap-persistence*
- Integration with Controls Configuration (CCS)
- Simplify configuration, hide JSON

DPP (offline)

- Technology research
  - Leveraging a rich Cloud-native (Kubernetes) landscape
  - Data Streaming solutions
- PoC Delivery: targeted for Q3 2025 including:
  - Two offline use cases from SigMon and OAF
  - EPA dashboards

MISC

- NXCALS/CCDB: Extend *logging-on-demand*
- More and better Python APIs (pyJapc → pyDA, …)

Summary

- There is a clear and growing need to provide a **D**ata **P**rocessing **P**latform as a Service
  - UCAP is successful and widely adopted for online use case
  - Multiple group in ATS have also implemented offline solutions their own way
- DPP is aimed to address both
  - It is endorsed by CTTB
  - Recruitment is ongoing
  - It factors in additional, emerging requirements, e.g., from EPA Work Packages
  - It brings more data in NXCALS, better Python support, better APIs
- **A Proof-of-Concept will be done in 2025 and we're looking forward to reporting back here in 1 year ☺**

Extra Slides

DPP - Ideas for the implementation

- User-Friendly Service Submission
  - Users can submit their code without worrying about the infra
  - Resources are scaled as needed
- Declarative usage: simplified interaction with existing systems (e.g., UCAP)
- Beginner friendly: prepare a simple analysis in less than 1 hour
  - Create a skeleton from a template → develop & test & run locally → deploy

```
→  knative dpp create -l python plotly-tmp-2
Created python function in /Users/msobiesz/dev/local/knative/plotly-tmp-2
→  knative cd plotly-tmp-2/
```

```
→  plotly-tmp-2 dpp deploy
   Function image built: gitlab-registry.cern.ch/msobiesz/knative-poc/plotly-tmp-2:latest
   Deploying function to the cluster
   Function deployed in namespace "default" and exposed at URL:
http://plotly-tmp-2.default.127.0.0.1.sslip.io
```

- Powered by a Kubernetes/Knative service, aka Function as a Service (FaaS)
  - Knative's Serving, Eventing, Func, Akka
- Plus: more powerful and convenient Python libraries to work with NXCALS
  - Pandas-on-Spark to run analysis in the NXCALS cluster;
  - API for cross-accelerator correlation (e.g. to follow an LHC bunch back to the PS cycle that produced it)
  - Local time by default (not UTC)

Online and offline – why and how to merge?

- **Having distinct solutions for online and offline is inefficient**
- **Online use case:**
  - **Scenario**: A UCAP transformation processes raw data from devices and publishes the results to NXCALS
  - **Advantage**: real-time availability of data e.g. for GUIs
  - **Downside**: the UCAP device must run continuously, else there are data losses
- **Corresponding offline use case as an alternative:**
  - **Scenario**: All raw data from devices is stored in NXCALS, and the same algorithm as above is executed in offline mode
  - **Advantage**: transformation can be run at anytime. If the algorithm changes, the whole year can be re-calculated
  - **Downside**: data cannot be displayed in real-time.
- **=> The two are complementary and should be combined.**

Challenges to achieve this: Finding a common way of expressing an algorithm for online and offline == common tools, data structures and APIs