



Accelerator Controls

Stephane Deghaye (BE-CSS)

14-03-2024

Agenda

1. **Control System Requirements**
2. **A bit of History**
3. **Hardware & Software Architecture**
4. **CERN examples & Key Components**

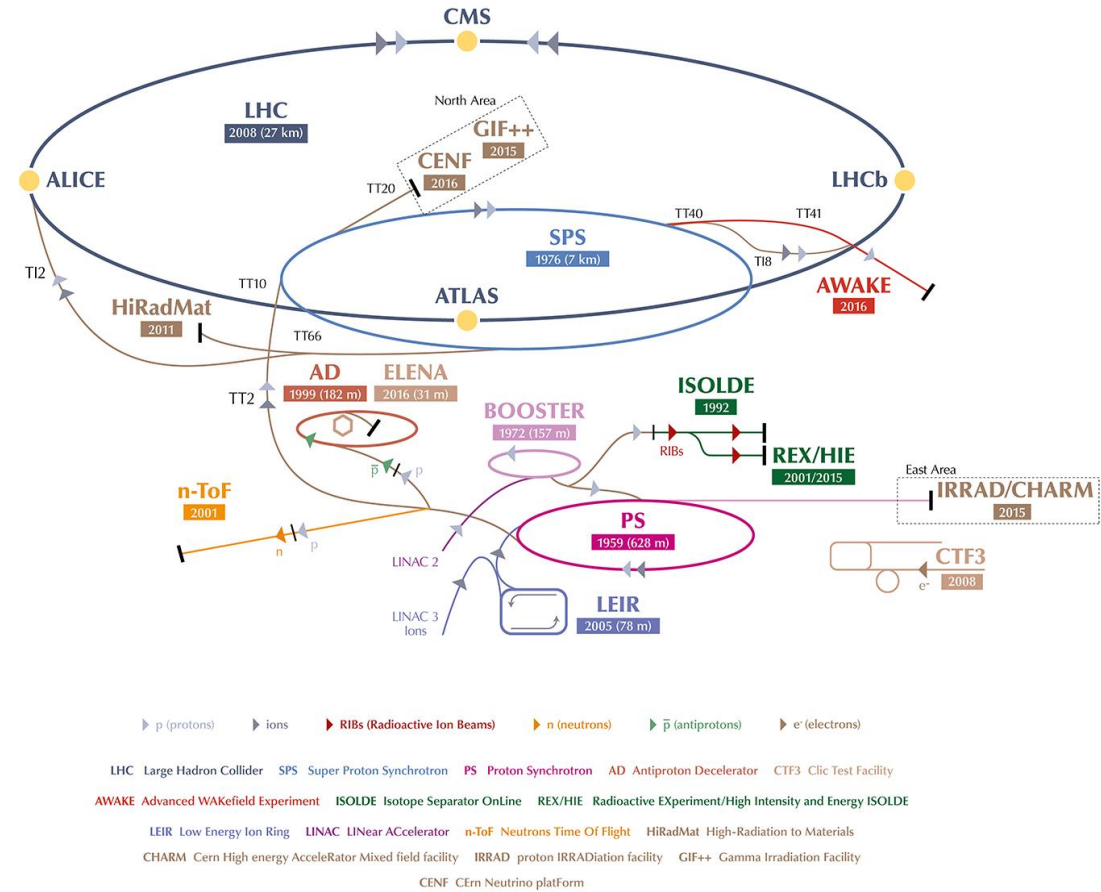
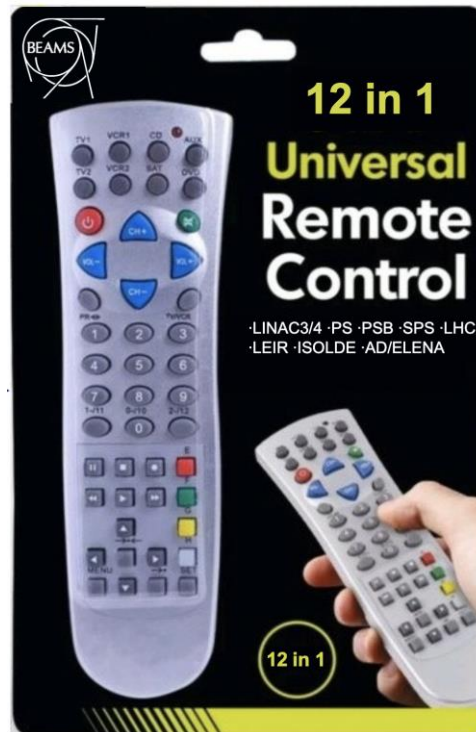
Why a Control System?

- The physicists and operators need to be able to remotely control and monitor the elements the particle accelerators are made off
 - ➔ this is the role of the **Control System**.
- The Control System's job is to provide to the physicists and operators a means to:
 - set reference values (aka setting) and states in active elements (e.g. power converters),
 - read instruments,
 - monitor the health of sub-systems,
 - diagnose faults,
 - etc.

Controls Complexity

- Many requirements from physicists and operators
- Accelerators made of many elements
 - Early accelerators, e.g. CERN Proton Synchrotron (PS), were small (< 5'000 devices)
 - Latest accelerators, e.g. LHC, are much more complex to operate (30'000+ elements)

The Control System's job is to hide the complexity and help you to do your job as efficiently as possible.

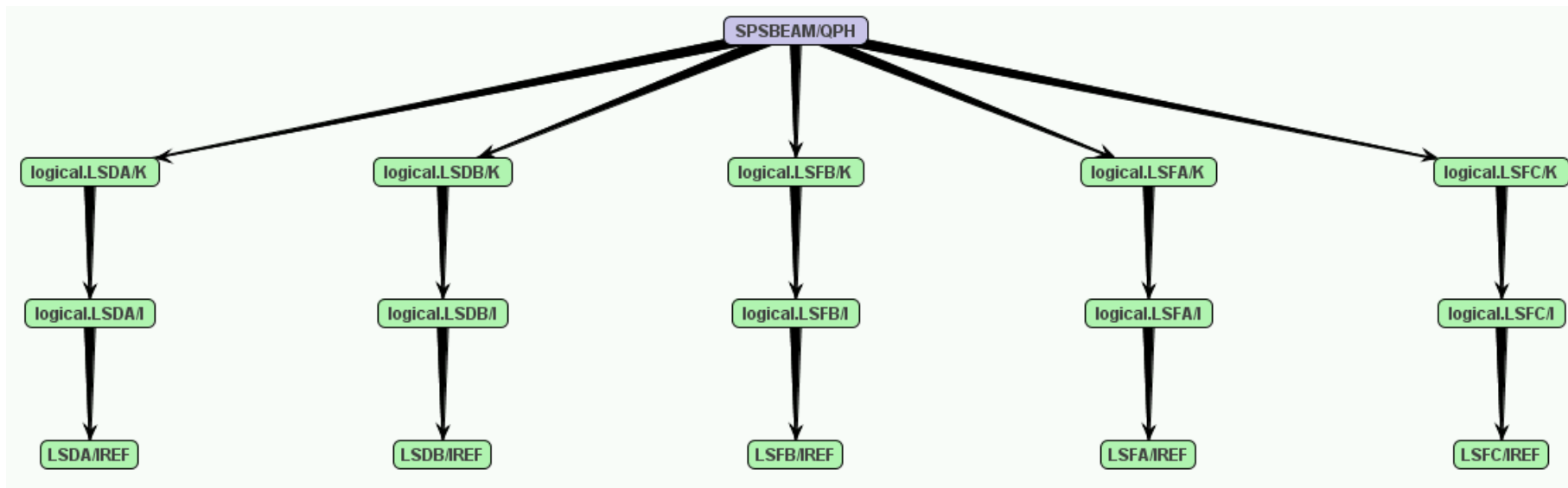


Control System Requirements

Control System Key Requirements

Act on accelerator elements (settings & states)

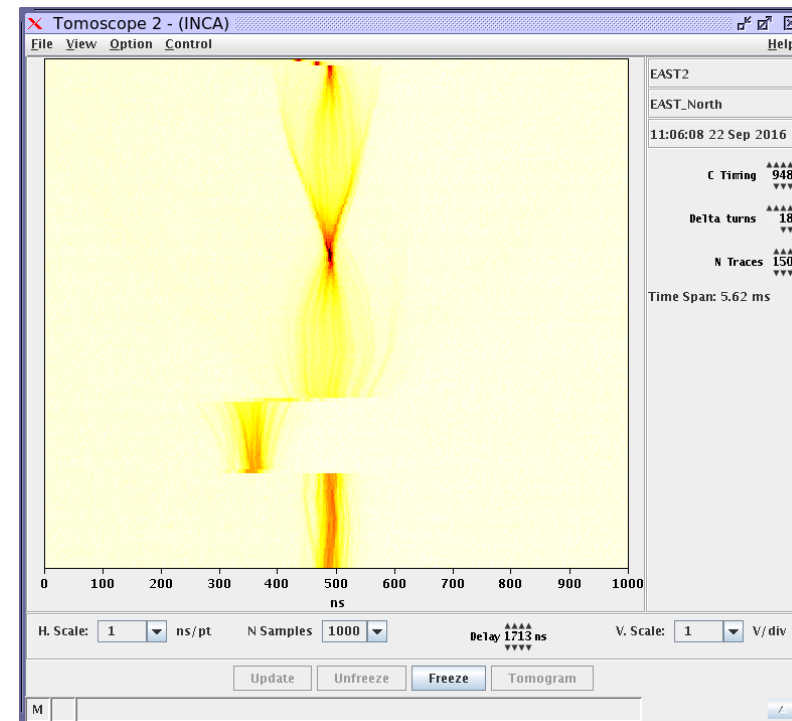
- **Minimum:** direct access to the hardware values
- **Ideal:**
 - **Model-driven control** to work at a higher level
 - **Global transactional synchronisation**



Control System Key Requirements

Monitor the elements (instruments & actuators)

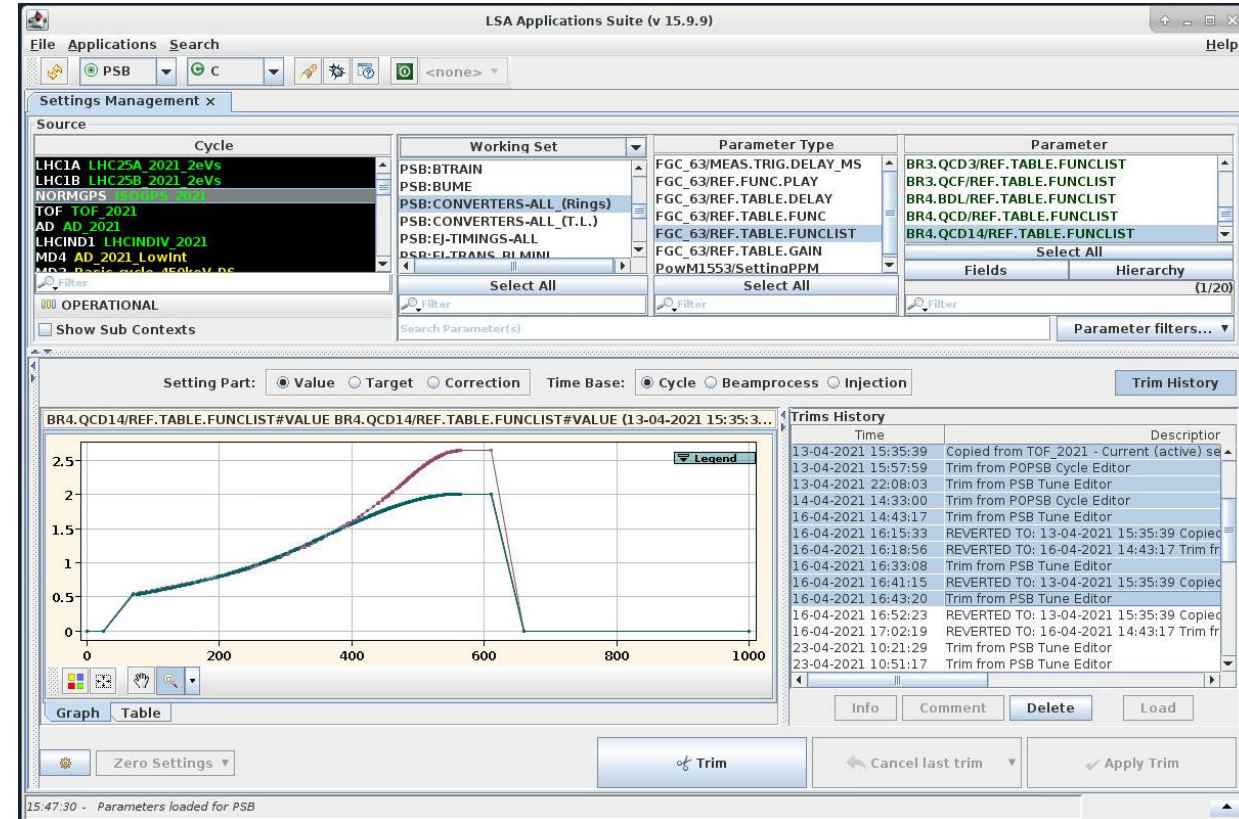
- **Minimum: Display raw acquisitions**
- **Ideal: Time-tagged, coherent acquisition, post-processing for quick detection of abnormal situations**



Control System Key Requirements

Long-Term Memory of Settings & Acquisitions

- AKA Logging
- Accelerator performance & post-mortem analysis, fine tuning of the machines
- Minimum: structured time-series in a simple format (CSV, SDDS, etc.)
- Ideal: Years of data (settings & acquisitions) with performant data extraction & analysis tools



More Control System Requirements

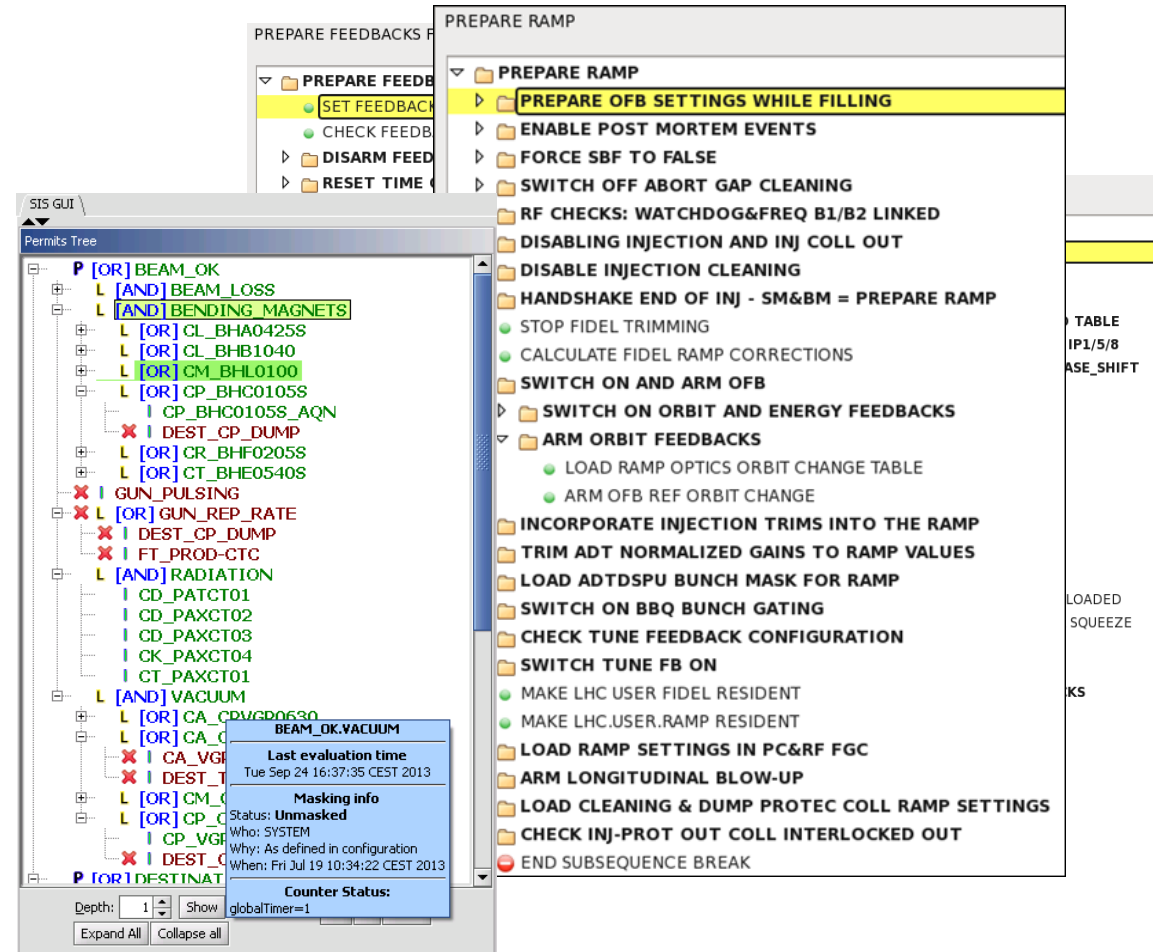
- **Safety for machine protection & operational availability**
 - **Minimum: Machine interlock to protect the hardware**
 - **Ideal: High-level fast-reaction interlocks and role-based access to prevent the wrong action at the wrong time**



More Control System Requirements

➤ Automation

- Generate initial values, play sequences, feedback loops, etc.
- Minimum: Non-interactive scripts
- Ideal: Model-driven generation, flexible sequencer (almost like a debugger), automated actions (decision tree, machine learning)



The image displays two overlapping windows from the SIS GUI. The background window shows a 'Permits Tree' with a hierarchical structure of logic gates and variables. The foreground window shows a 'PREPARE RAMP' sequence of actions.

Permits Tree (Background):

- P [OR] BEAM_OK
 - L [AND] BEAM_LOSS
 - L [AND] BENDING_MAGNETS
 - L [OR] CL_BHA0425S
 - L [OR] CL_BHB1040
 - L [OR] CM_BHL0100
 - L [OR] CP_BHC0105S
 - I CP_BHC0105S_AQN
 - X I DEST_CP_DUMP
 - L [OR] CR_BHF0205S
 - L [OR] CT_BHE0540S
- X I GUN_PULSING
- X L [OR] GUN_REP_RATE
 - I DEST_CP_DUMP
 - I FT_PROD-CTC
- L [AND] RADIATION
 - I CD_PATCT01
 - I CD_PAXCT02
 - I CD_PAXCT03
 - I CK_PAXCT04
 - I CT_PAXCT01
- L [AND] VACUUM
 - L [OR] CA_CPVGD0630
 - L [OR] CA_C...
 - X I CA_VGI...
 - X I DEST_...
 - L [OR] CM_C...
 - L [OR] CP_C...
 - I CP_VGI...
 - X I DEST_C...
- P [OR] DESTINAT...

PREPARE RAMP (Foreground):

- PREPARE FEEDBACKS
- PREPARE FEEDBACKS
 - SET FEEDBACKS
 - CHECK FEEDBACKS
 - DISARM FEEDBACKS
 - RESET TIME
- PREPARE RAMP
 - PREPARE OFB SETTINGS WHILE FILLING
 - ENABLE POST MORTEM EVENTS
 - FORCE SBF TO FALSE
 - SWITCH OFF ABORT GAP CLEANING
 - RF CHECKS: WATCHDOG&FREQ B1/B2 LINKED
 - DISABLING INJECTION AND INJ COLL OUT
 - DISABLE INJECTION CLEANING
 - HANDSHAKE END OF INJ - SM&BM = PREPARE RAMP
 - STOP FIDEL TRIMMING
 - CALCULATE FIDEL RAMP CORRECTIONS
 - SWITCH ON AND ARM OFB
 - SWITCH ON ORBIT AND ENERGY FEEDBACKS
 - ARM ORBIT FEEDBACKS
 - LOAD RAMP OPTICS ORBIT CHANGE TABLE
 - ARM OFB REF ORBIT CHANGE
 - INCORPORATE INJECTION TRIMS INTO THE RAMP
 - TRIM ADT NORMALIZED GAINS TO RAMP VALUES
 - LOAD ADTSPU BUNCH MASK FOR RAMP
 - SWITCH ON BBQ BUNCH GATING
 - CHECK TUNE FEEDBACK CONFIGURATION
 - SWITCH TUNE FB ON
 - MAKE LHC USER FIDEL RESIDENT
 - MAKE LHC.USER.RAMP RESIDENT
 - LOAD RAMP SETTINGS IN PC&RF FGC
 - ARM LONGITUDINAL BLOW-UP
 - LOAD CLEANING & DUMP PROTEC COLL RAMP SETTINGS
 - CHECK INJ-PROT OUT COLL INTERLOCKED OUT
 - END SUBSEQUENCE BREAK

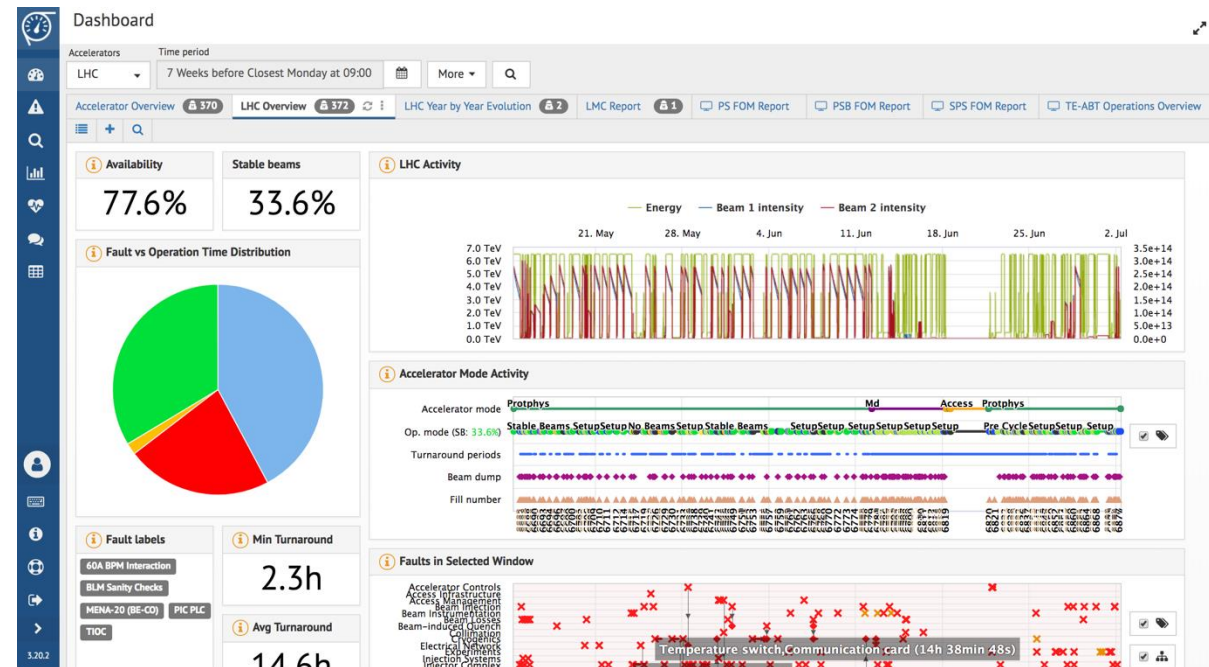
BEAM_OK.VACUUM Pop-up:

- Last evaluation time: Tue Sep 24 16:37:35 CEST 2013
- Masking info: Status: Unmasked, Who: SYSTEM, Why: As defined in configuration, When: Fri Jul 19 10:34:22 CEST 2013
- Counter Status: globalTimer=1

More Control System Requirements

➤ Diagnostics

- Detection, identification, and follow-up of problems in the controls infrastructure
- Minimum: Non-interactive status screens
- Ideal: Online monitoring, remote interventions (e.g. power cycle), failure prediction (Machine Learning), analysis tools



And many more...

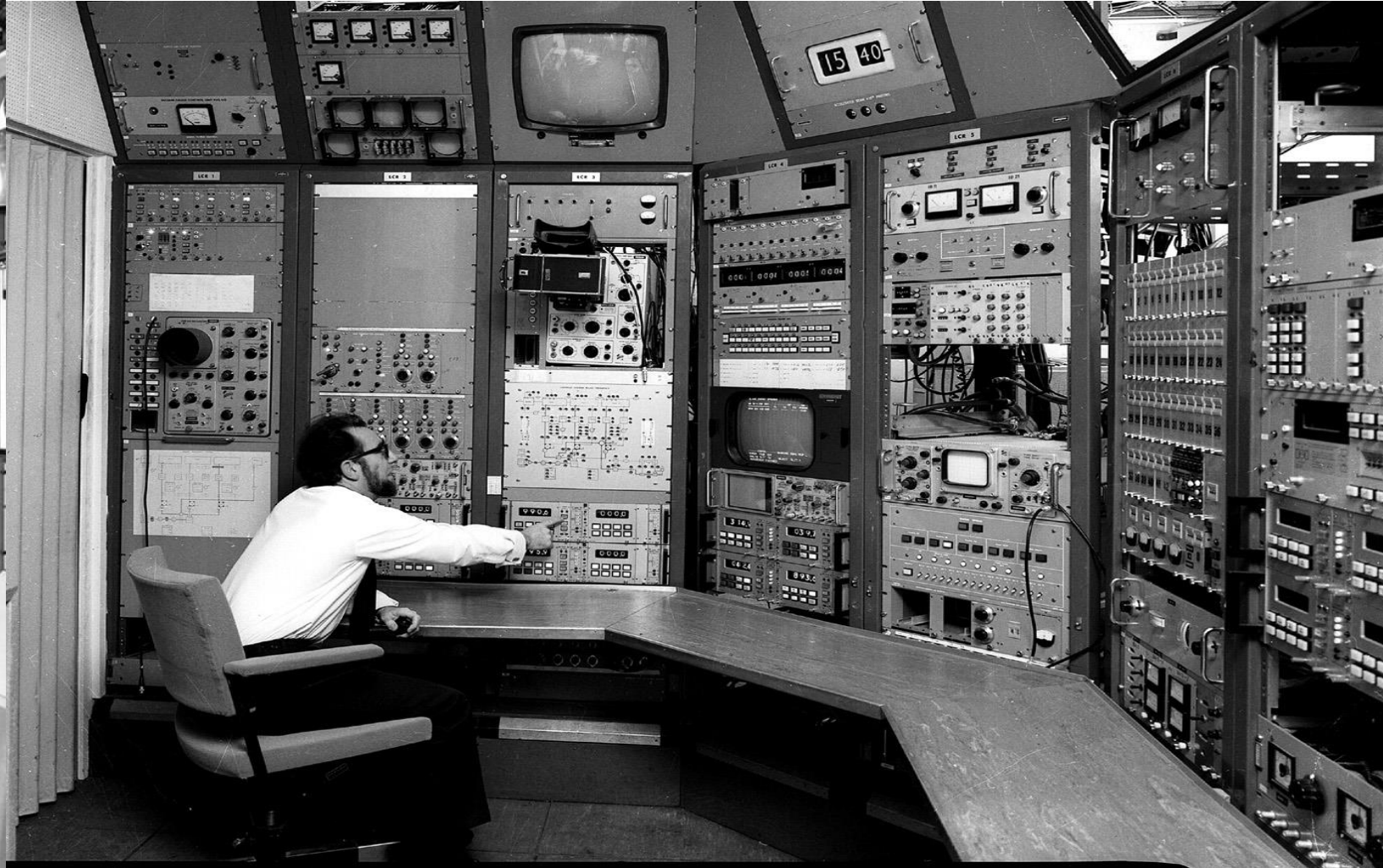


A bit of History

Control System Prehistory

- **Accelerators are small and overall less complex (e.g. no superconducting magnets)**
 - No more than a few thousands of devices to control
- **No computing infrastructure and limited possibility to model**
- **Actuator and monitors are physically in the local control rooms (e.g. buttons, knobs, analogue oscilloscopes, etc.)**

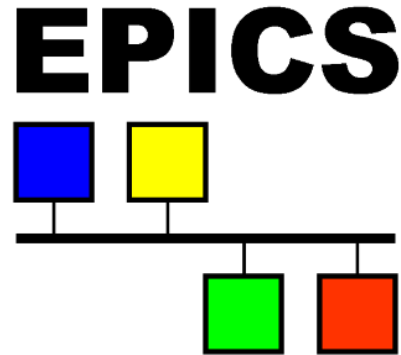
Control System Prehistory



The Early Days

Beginning of remote controls

- Still limited by the available performance
- Lack of standards and common frameworks → more DIY and custom solutions
- Emergence of several controls solutions, aiming at different types of accelerators (at first)
 - EPICS (driven by US labs),
 - Tango (driven by ESRF (Fr) – synchrotron light sources)
 - CERN¹



¹ non-exhaustive list

Modern days

➤ Hardware has become powerful

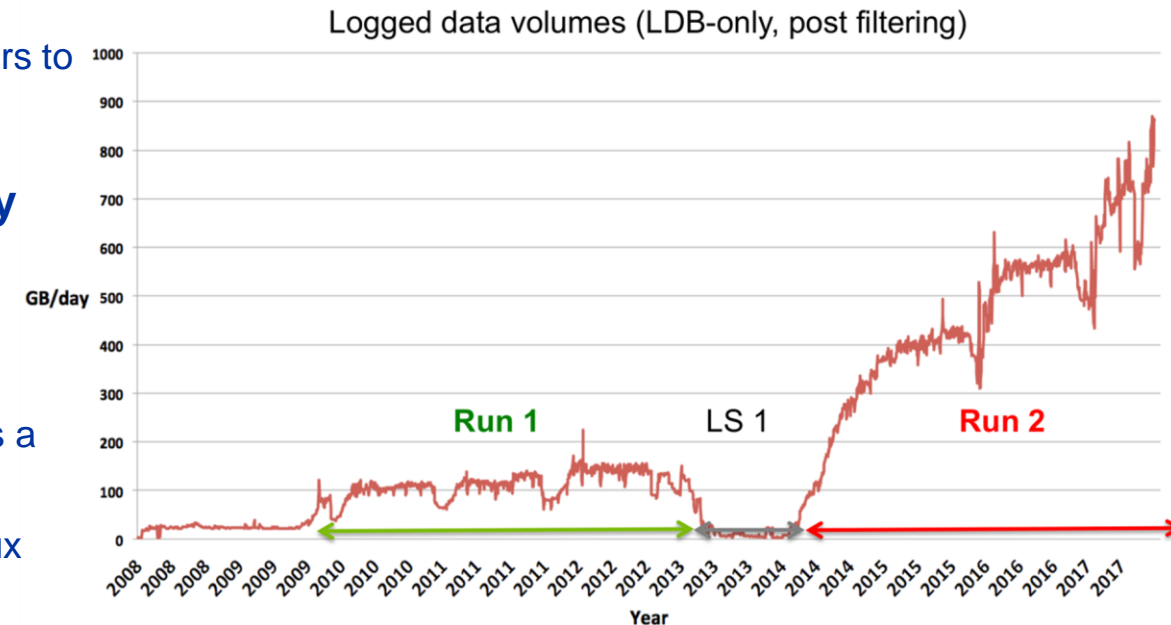
- E.g. embedded systems at CERN in late 90s had **64 MB of RAM**; Nowadays, they have **8 GB**
- Most of the needs are covered
- Yet, users want more and more data (turn-by-turn acquisitions, big-data solution for the long-term storage, etc.)

➤ Software industry has become a major actor worldwide.

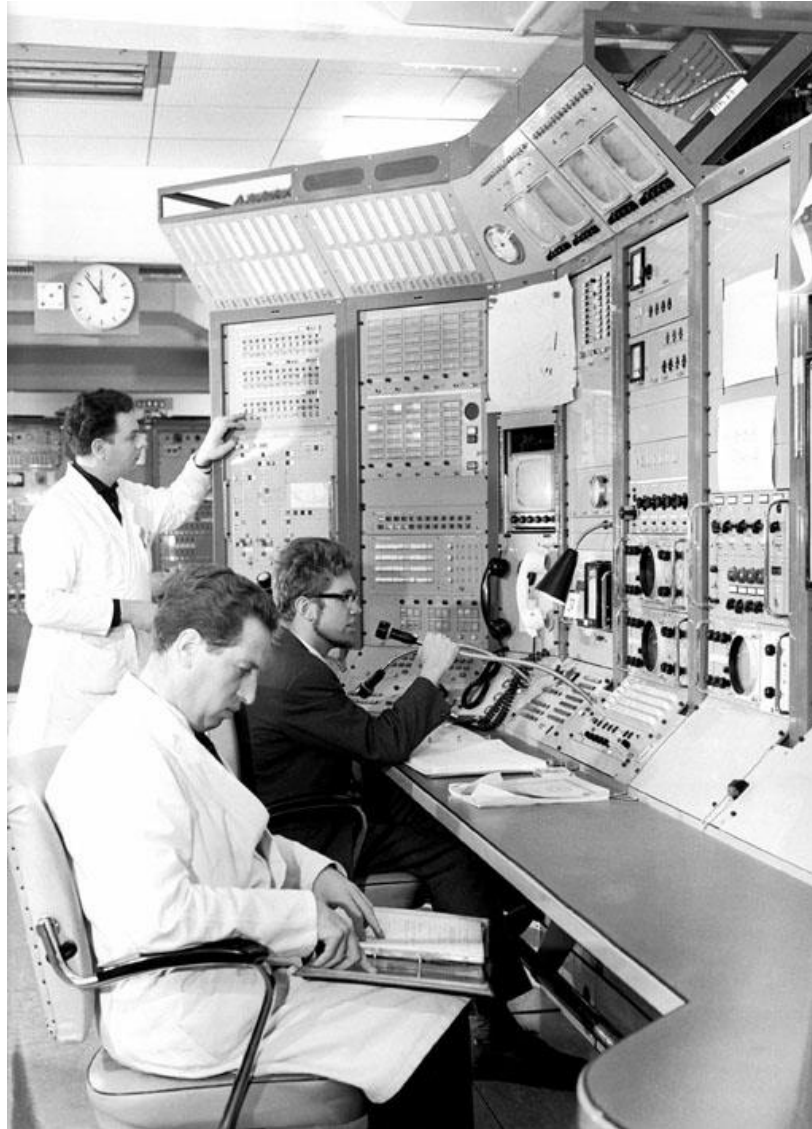
- We can rely on many readily available technologies that open the doors to much more powerful systems

➤ We still need to integrate and customise them to the very specific domain of particle accelerator controls

- Not all solutions are appropriate; Need to remember accelerator controls \neq selling plane tickets
- Mastering the different solutions with their evolution, limitations, etc. is a major challenge
- The rhythm of updates is no longer under our control. E.g. recent Linux CentOS changes



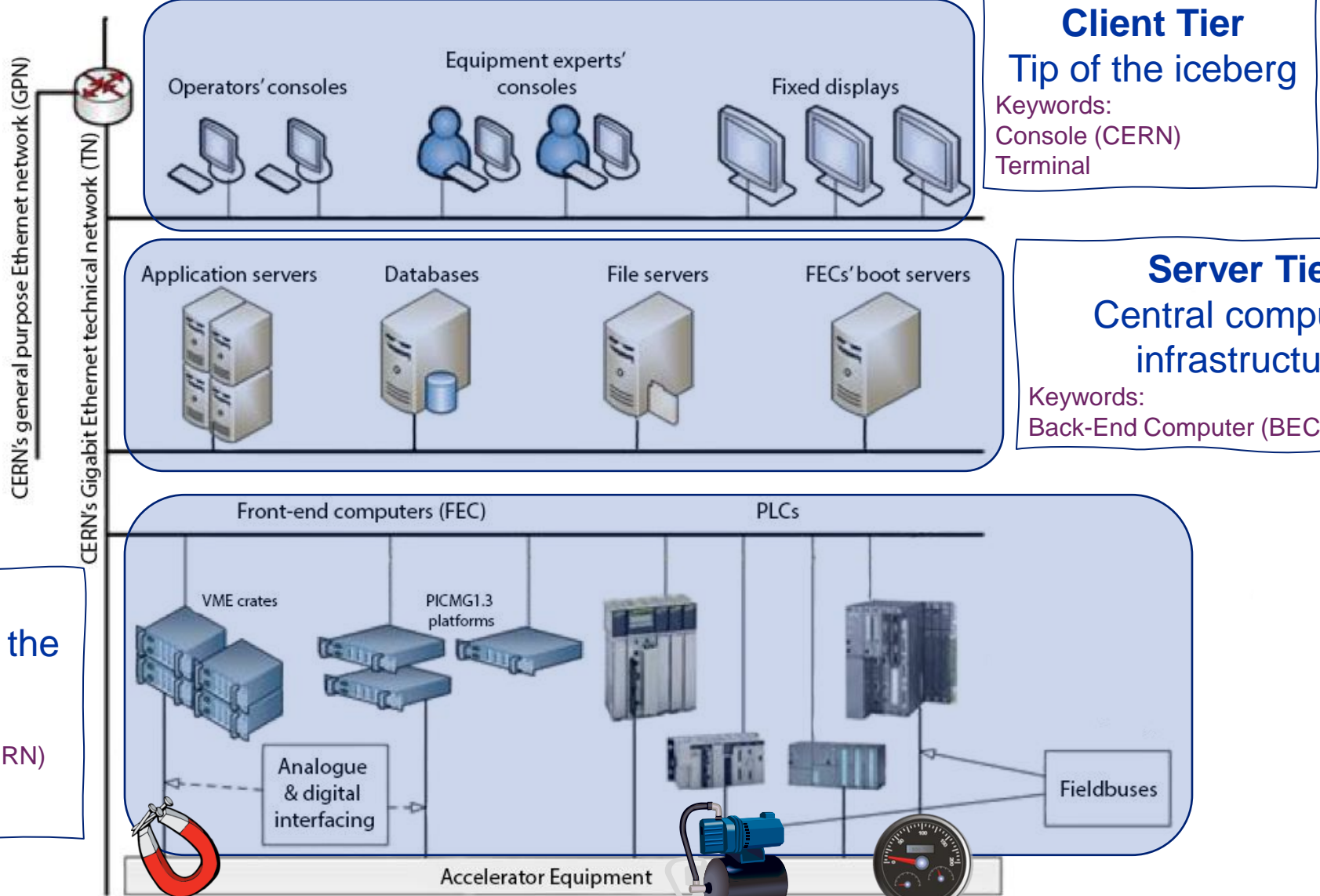
50 years of technology evolution



High-Level Architecture

of a modern control system

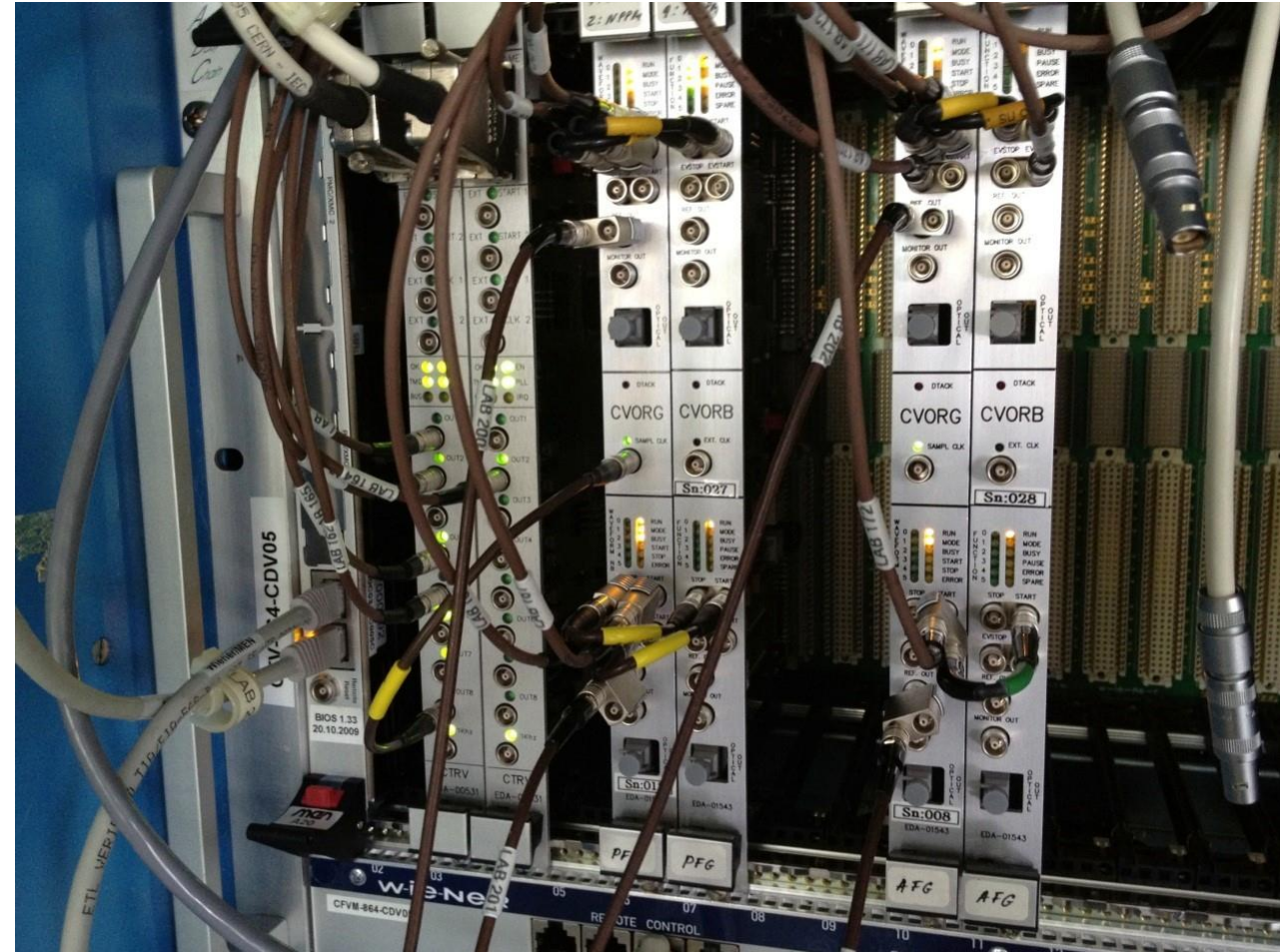
High-Level Hardware Architecture



High-Level Hardware Architecture

Resource Tier

- **Open enclosures**
 - Easy access
 - Better cooling and power available
 - But expensive
- **Closed enclosures**
 - Possible for simple functions (e.g. fieldbus control)
 - Cost effective; when deployed in big number, e.g. LHC power converter control gateways



High-Level Hardware Architecture

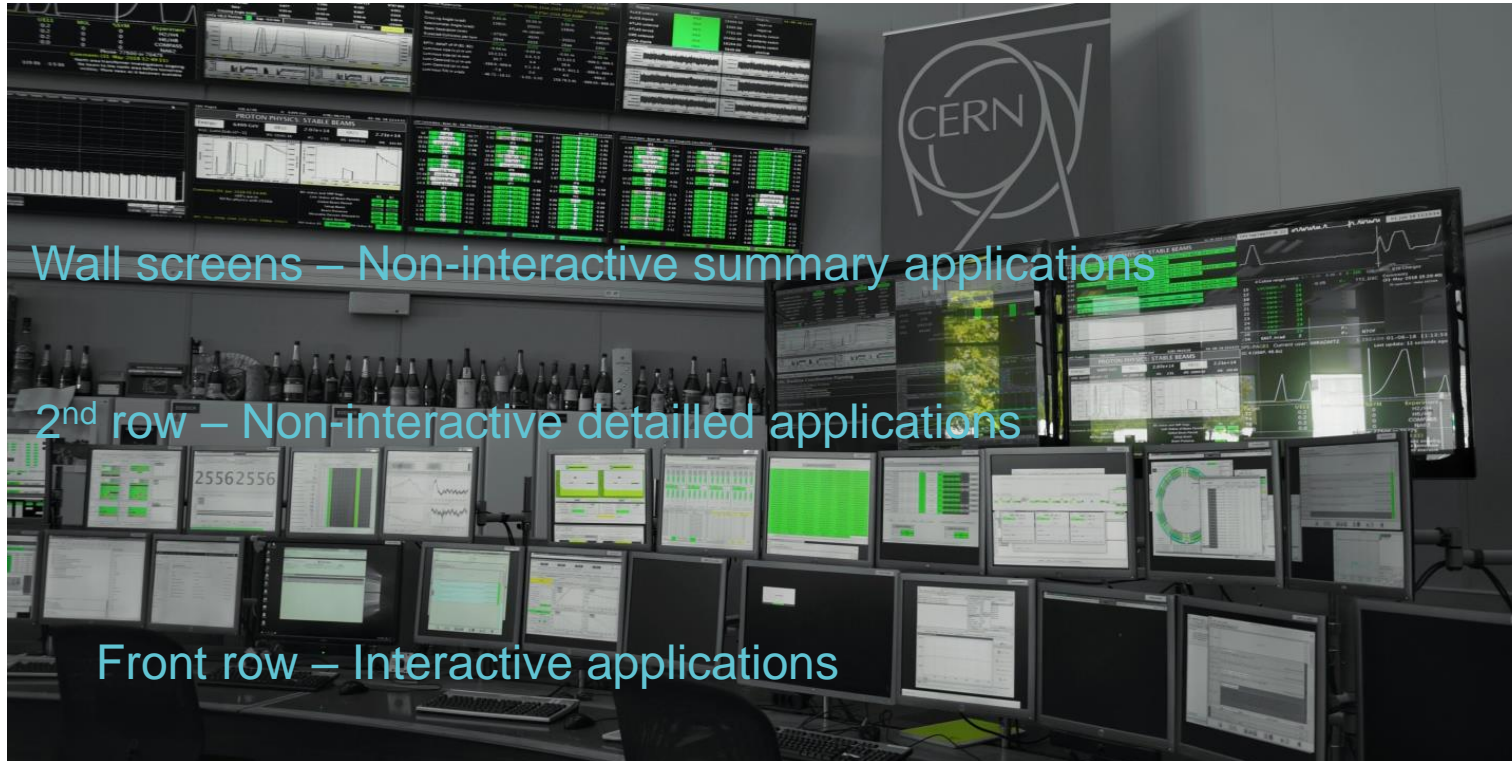


Middle Tier

- IT Computer centre type of hardware
- High-density
- Highly available (redundancy and hot-swap)



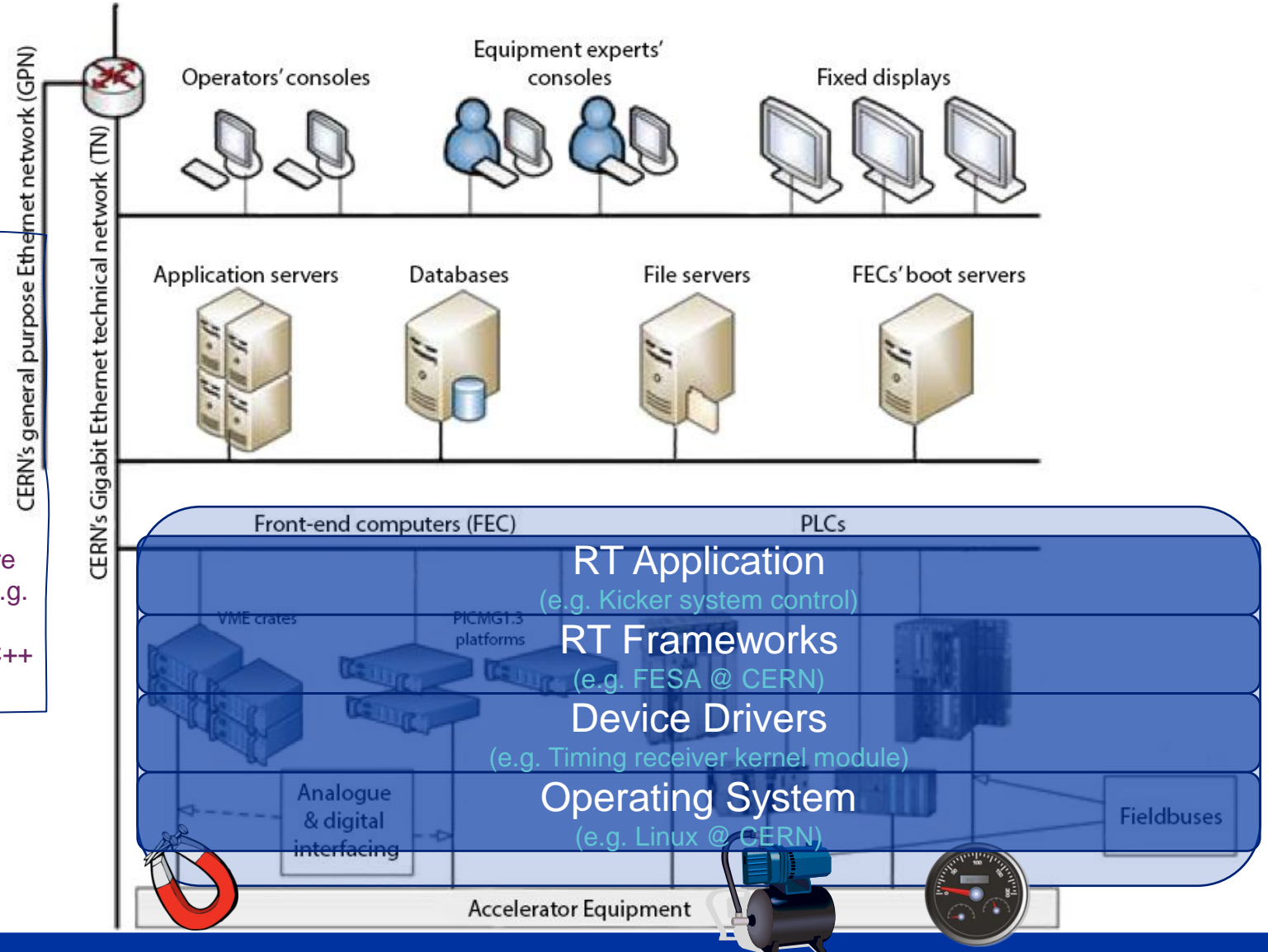
High-Level Hardware Architecture



Control Room Computers

- As much as possible COTS desktop PCs but MTBF requirements might be difficult to satisfy
- Users expect modern reactive GUIs
- Several layers of screens to have as much data as possible available

High-Level Software Architecture



Front-end Tier

Real-time control and acquisition

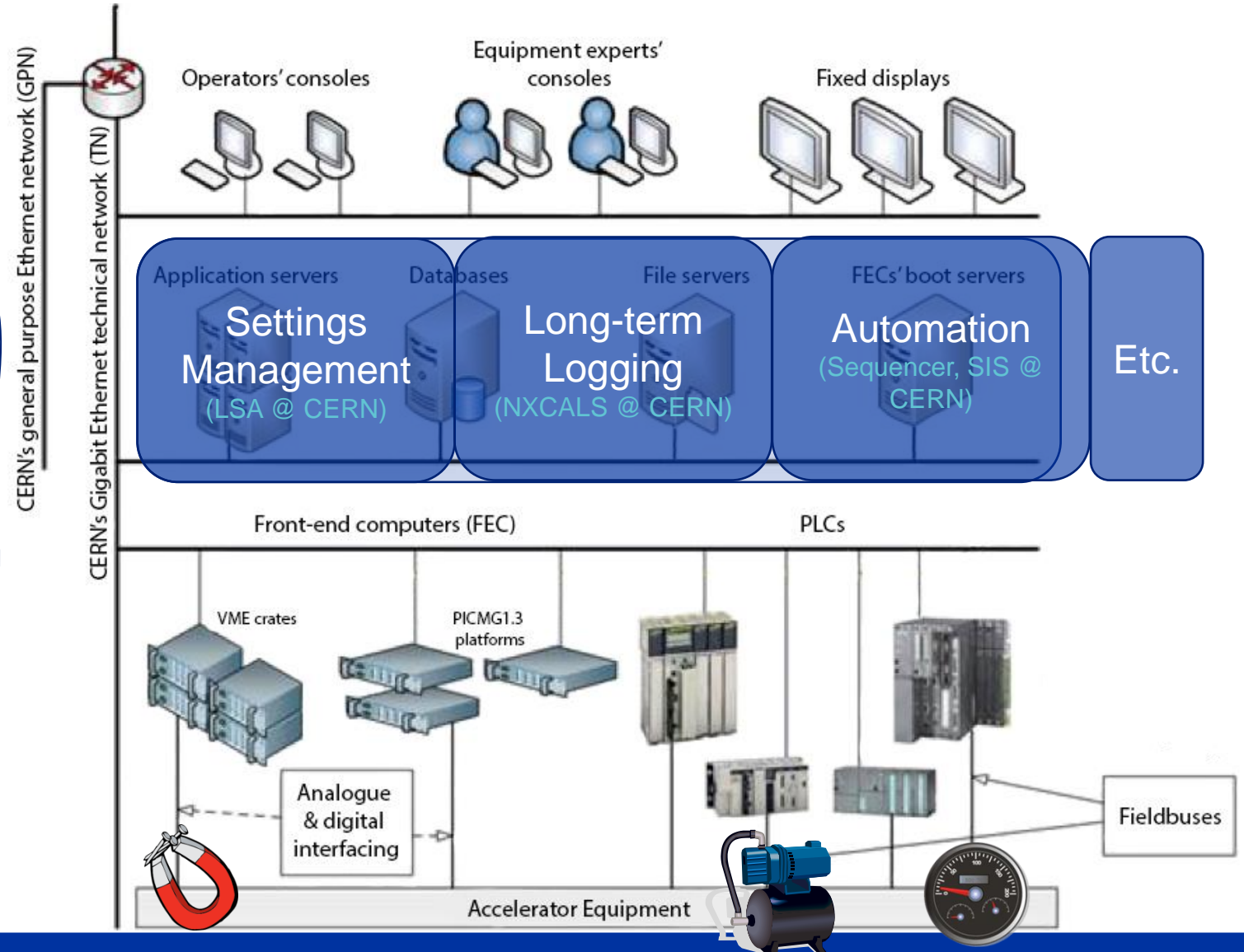
- Limited, local scope
- Fast reaction possible (interrupts)
- Limited computing power (compared to other tiers)
- Equipment processing to provides a high-level view of the hardware
- Real-time (RT) applications relies on frameworks, which capture the recurring aspects (react to events, publish new data, etc.) E.g. FESA @ CERN, POGO with Tango
- Based on technologies closed to the hardware (C for drivers, C++ for RT, etc.)

High-Level Software Architecture

Business Tier

General purpose services & Specific business logic

- Broader scope; able to coordinate the entire accelerator
- Powerful computers
- Less reactive (network) and at a higher-level of abstraction
- Based on technologies that are better suited for high-level business logic (e.g. Java)



High-Level Software Architecture

Presentation Tier Graphical applications

Different technologies available

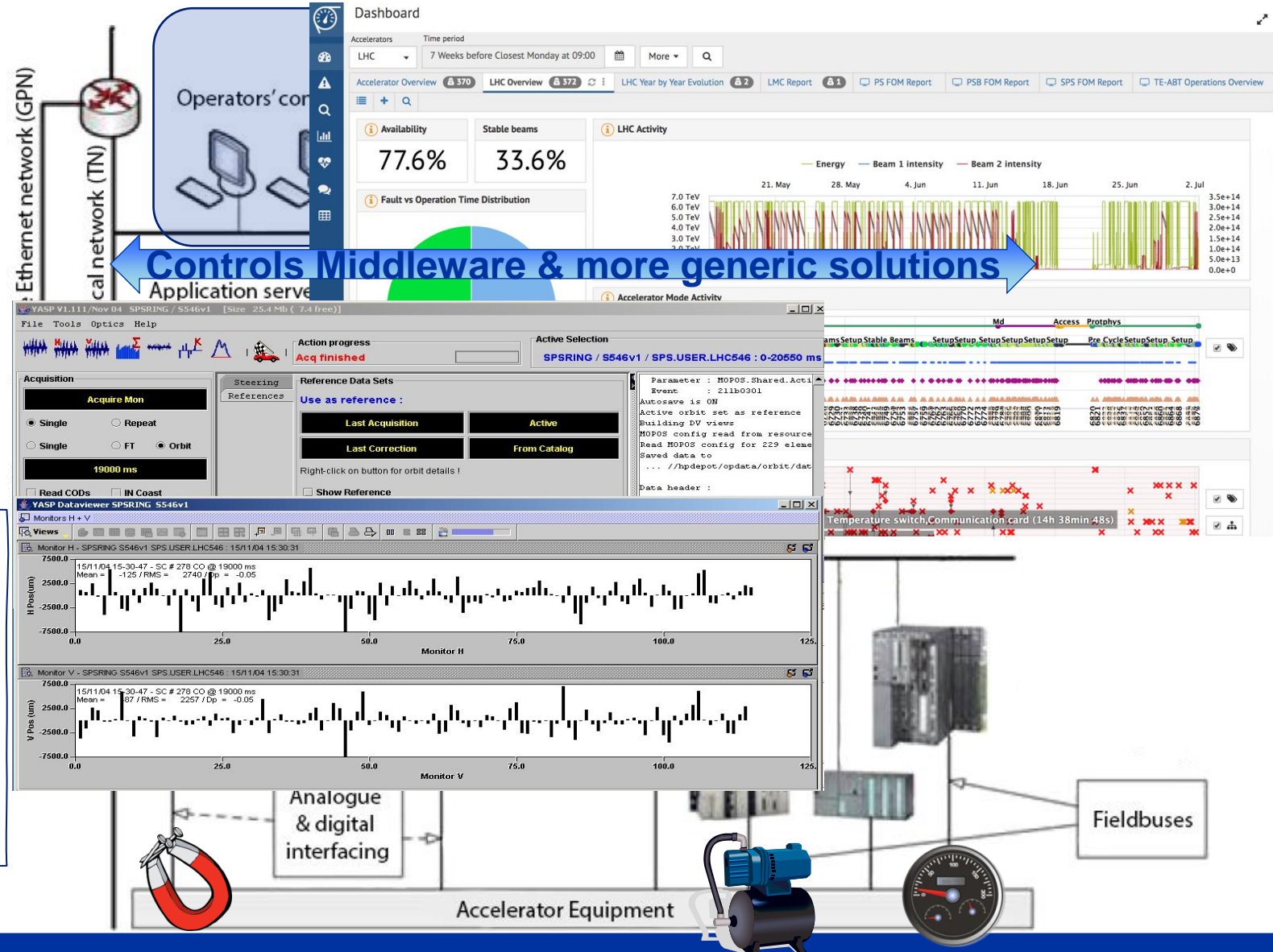
- Python with PyQt
- Web ecosystem (**Angular**, View.js, etc.)
- Java Swing, Java FX

Keywords:

- Graphical User Interface (GUI)
- Human-Machine Interface (HMI)
- Command-line interface (CLI)

Communication

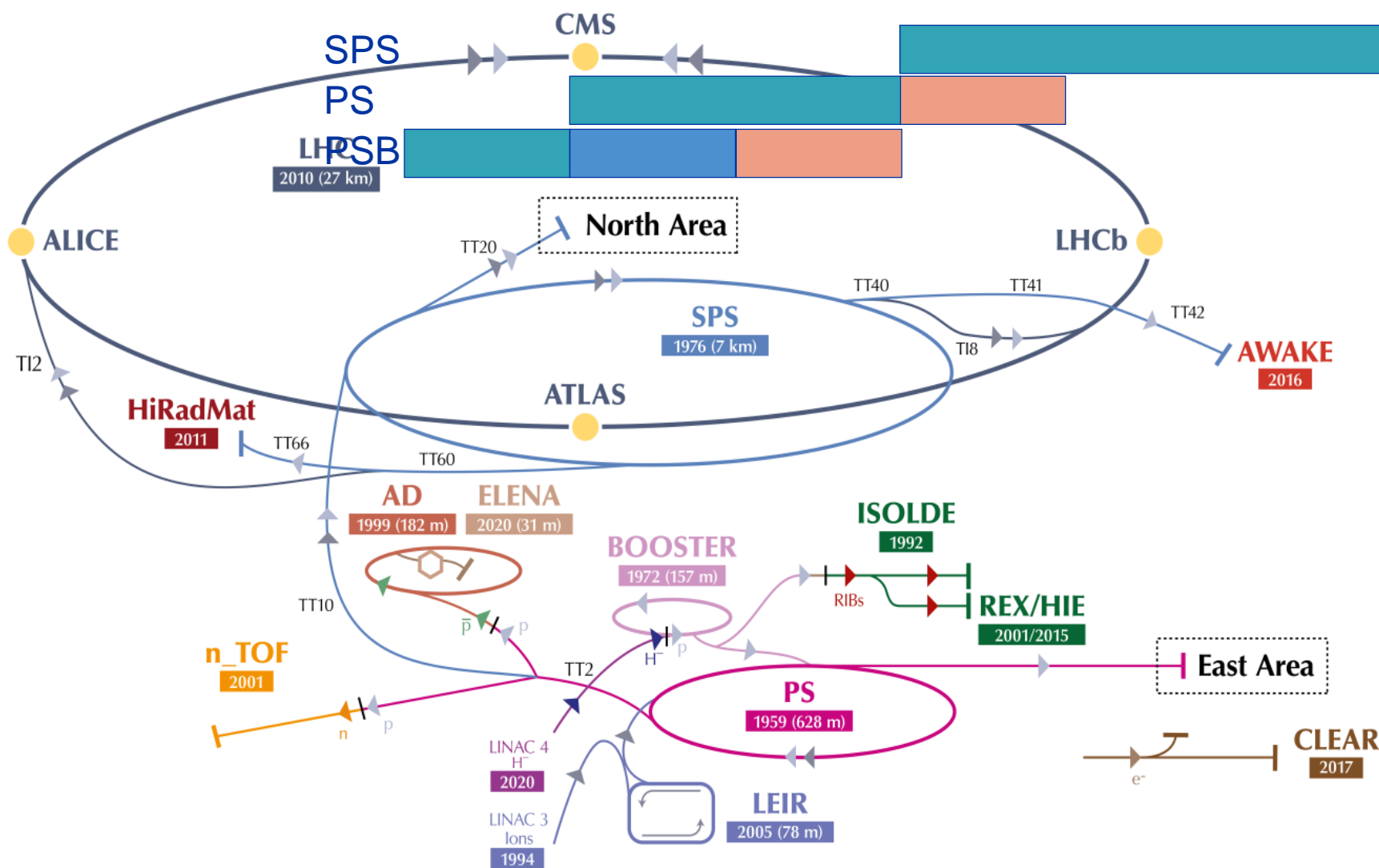
- Accelerator-specific protocols for the lower layers
 - Channel Access (EPICS)
 - CMW (CERN)
- Potentially, more generic technologies for the higher layers
 - RMI/JMS
 - REST API
 - gRPC
 - ...



A few examples

And the key components used...

Accelerators Optimisation

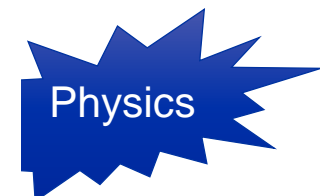


Optimise the throughput of the accelerators

Requirements: Change references of most all equipment from pulse to pulse

The control system must be hard real-time

The central timing system sequences the entire beam production

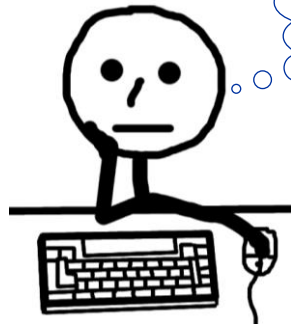


Use Case 1: Control

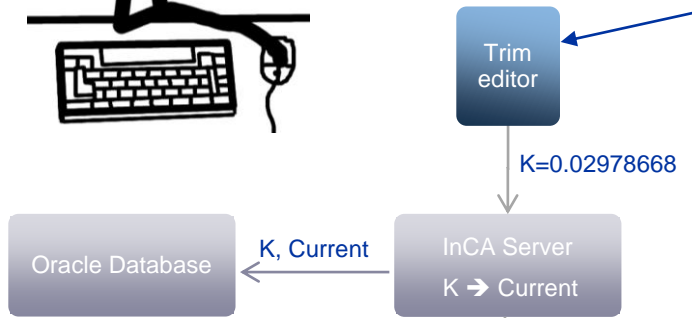
- **Who:** PS operator
- **What:** Change the strength of the extraction septum towards the SPS for the Fixed Target beam
- **Involved controls components:**
 - InCA/LSA (Setting management)
 - CMW (Controls Middleware)
 - FESA (Real-time hardware control)
 - Timing (Synchronisation)

In all the examples, we assume all the configuration is already done and we focus on the run-time aspects

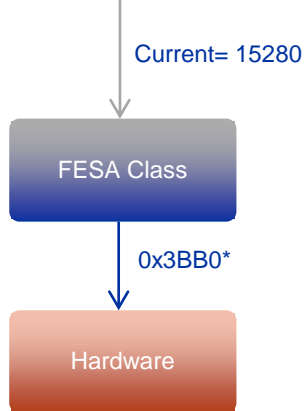
Example 1 – Control



More strength on my extraction septum → 0.02978668



Control now!

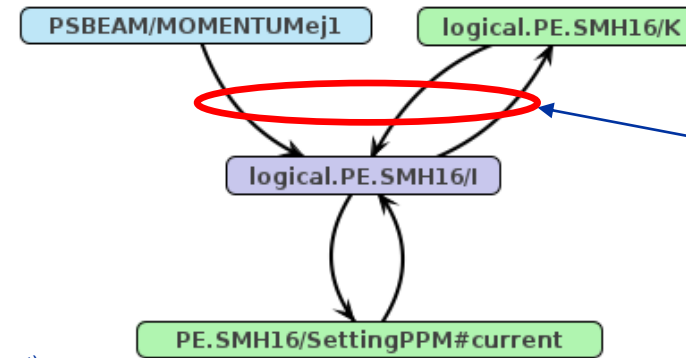
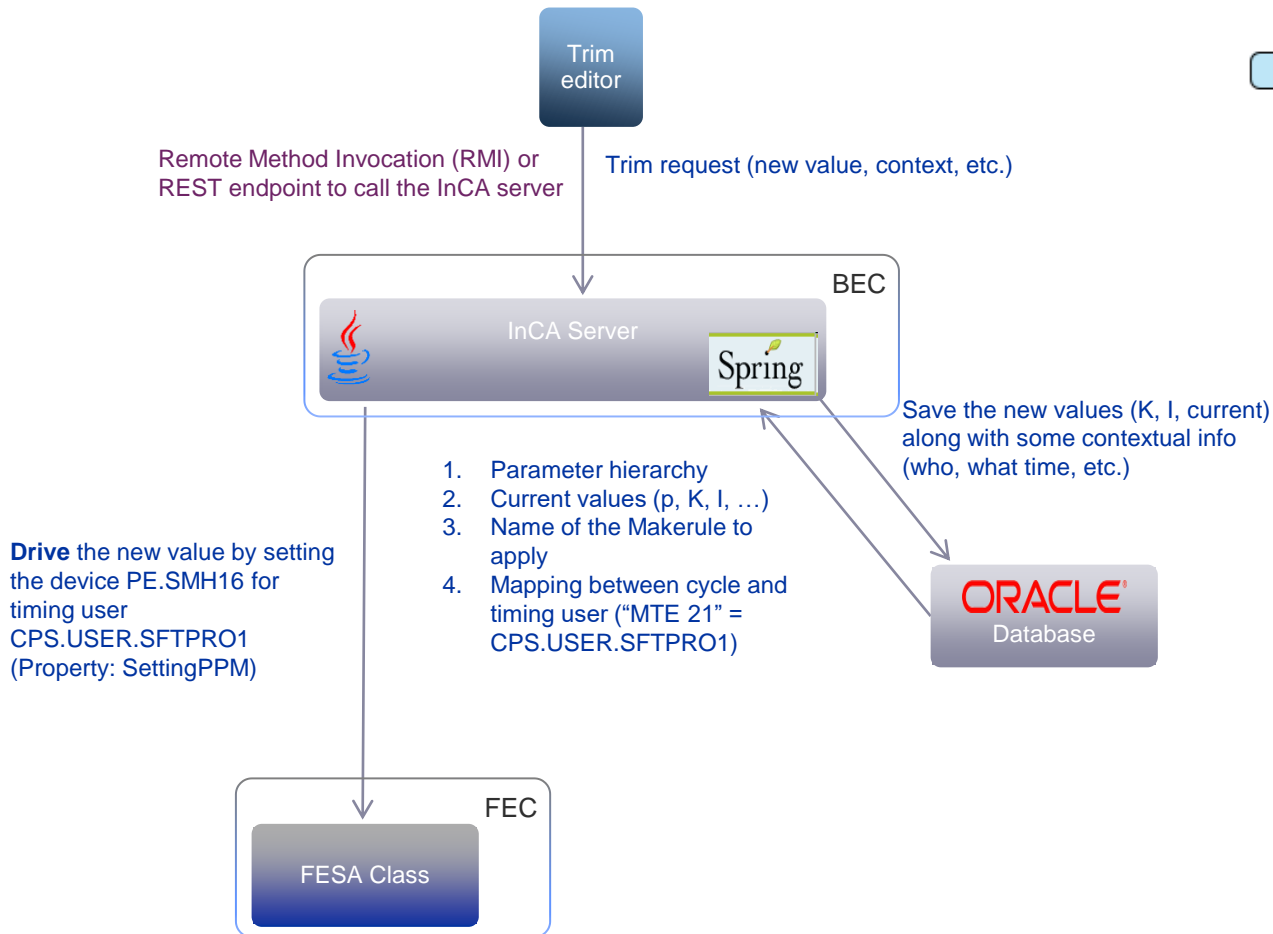


Timing System (GMT)

The screenshot shows the 'Settings Management' window in the LSA Applications Suite. The 'Parameter' column lists several parameters, with 'logical.PE.SMH16/K' circled in red. Below the list, the 'Trim' button is also circled in red. The 'Trim editor' window shows the value '0.02978668' for the parameter 'logical.PE.SMH16/K'.

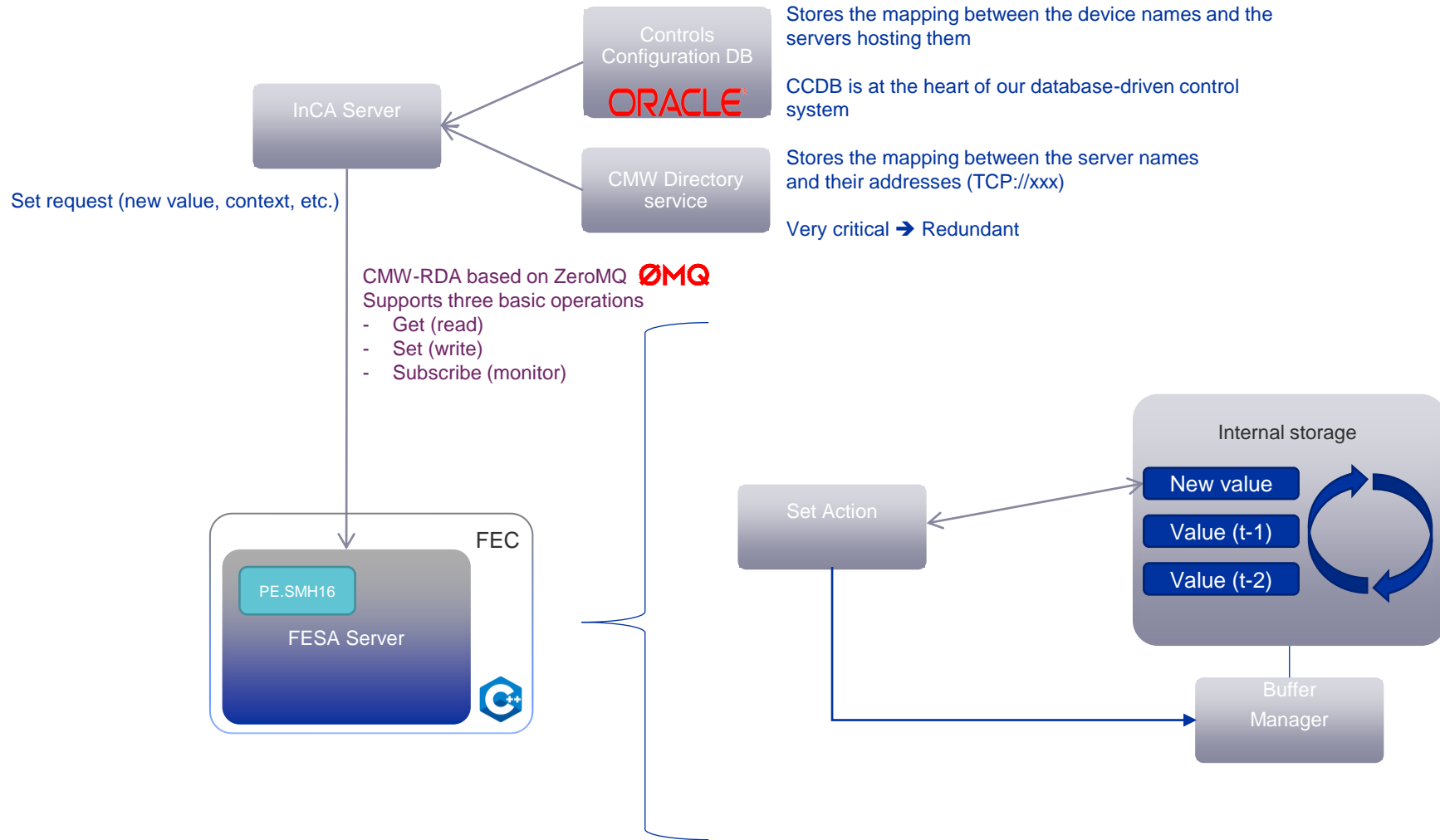
*not a real value

Example 1 – Control - Communication

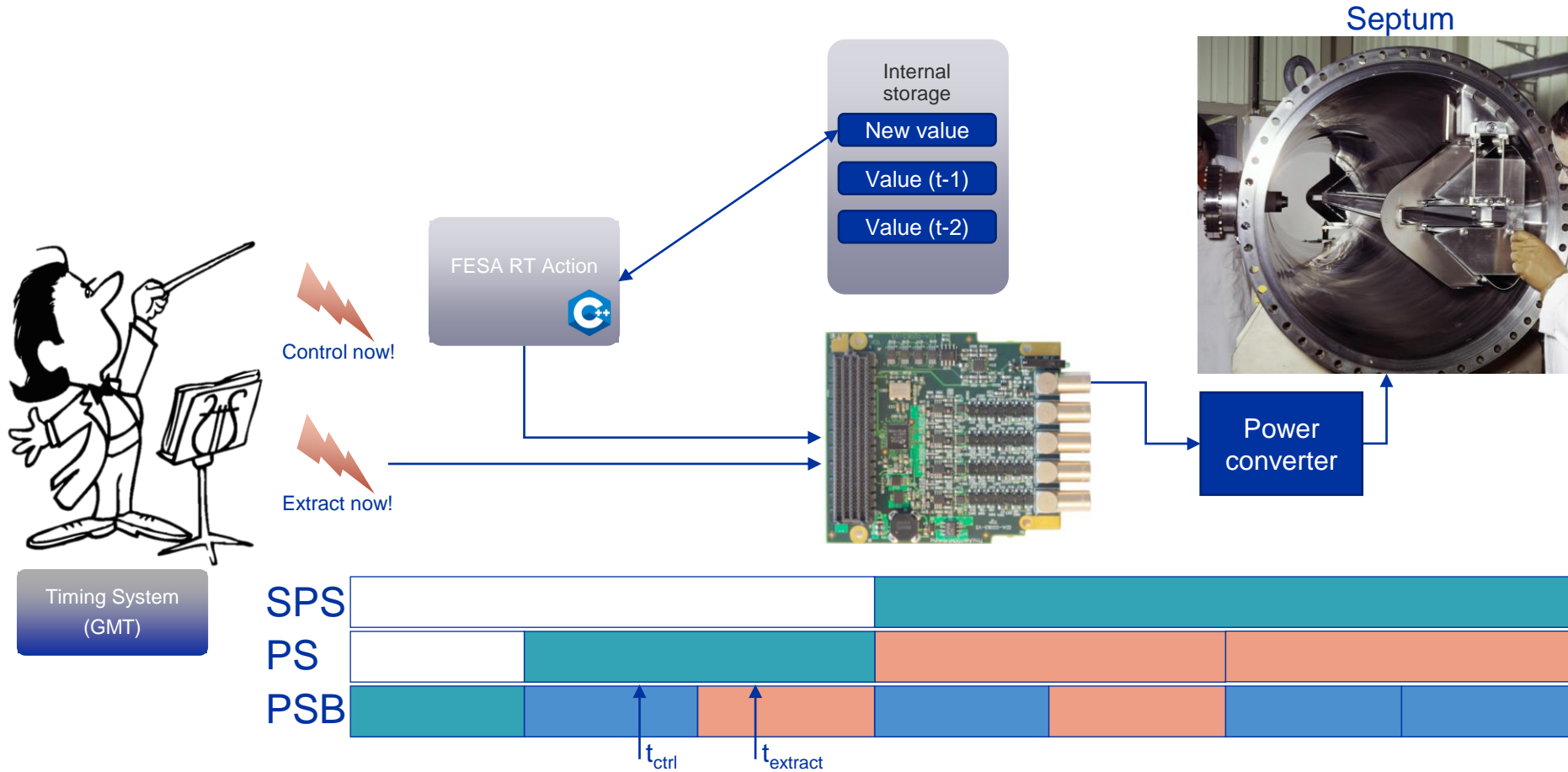


Makerule: Java class to compute parameters based on other parameters, hardware characteristics (calibration curve) and cycle specific values

Example 1 – Control – Low-level



Example 1 – Control – Beam production

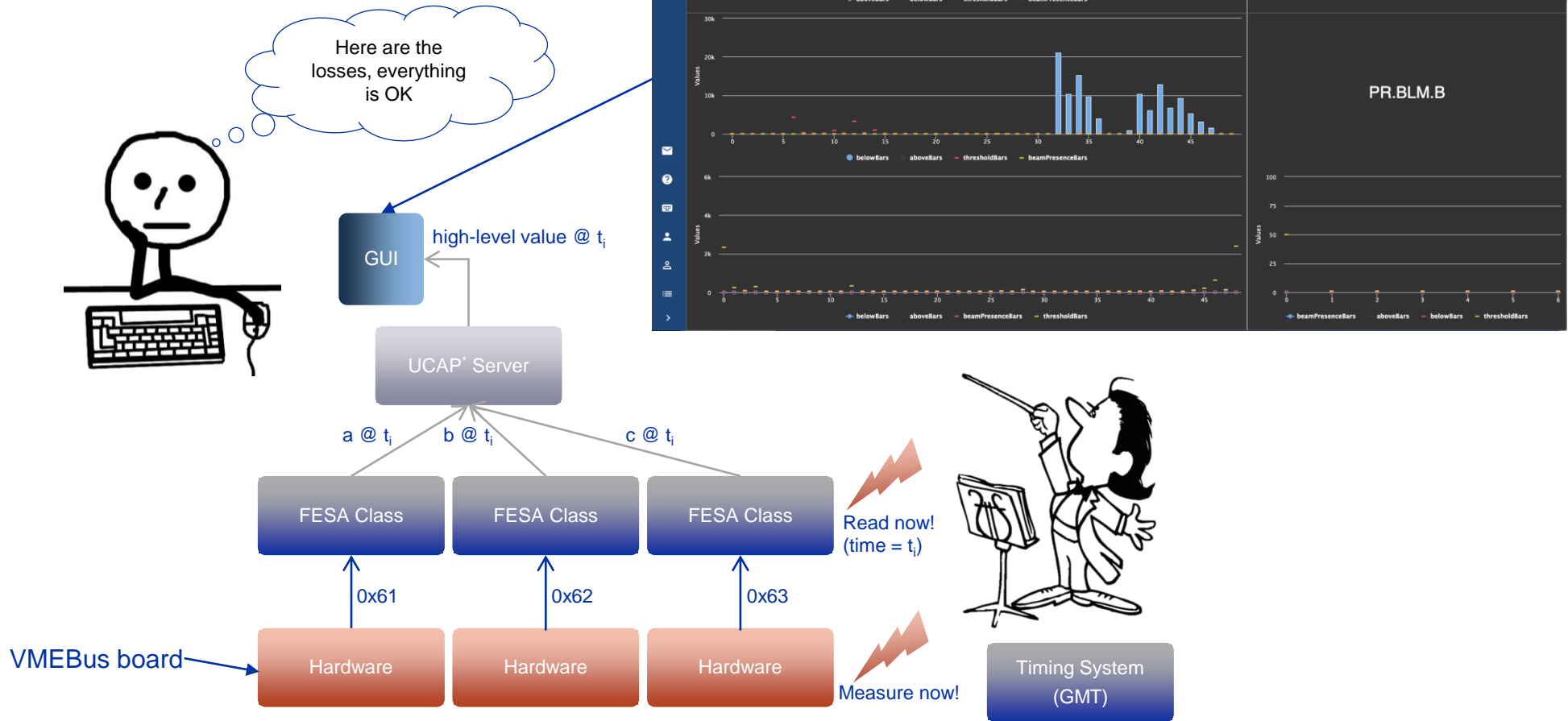


Use Case 2: Acquisition

- **Who:** PS operator
- **What:** Keep an eye on acquisition values of many control devices. The low-level data needs post-processing and will be displayed as a graph in a web page
- **Involved controls components:**
 - Timing (Synchronisation)
 - FESA (Real-time hardware control)
 - CMW (Controls Middleware)
 - UCAP (Unified Controls Acquisition & Processing)
 - WRAP (Web Rapid Application Platform)

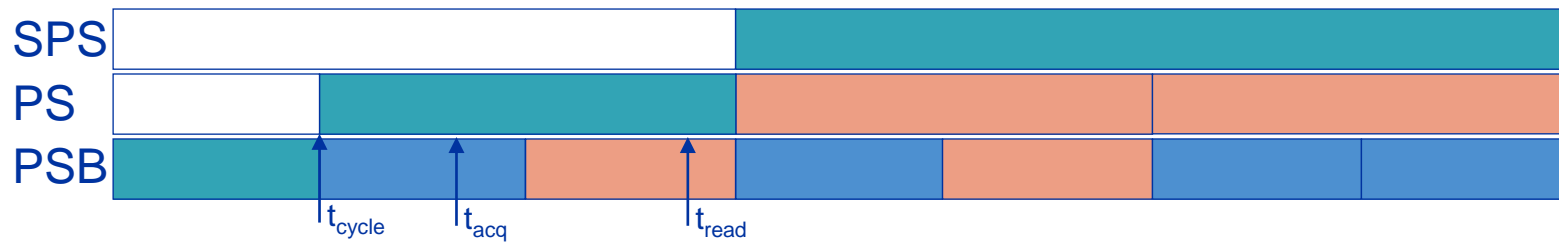
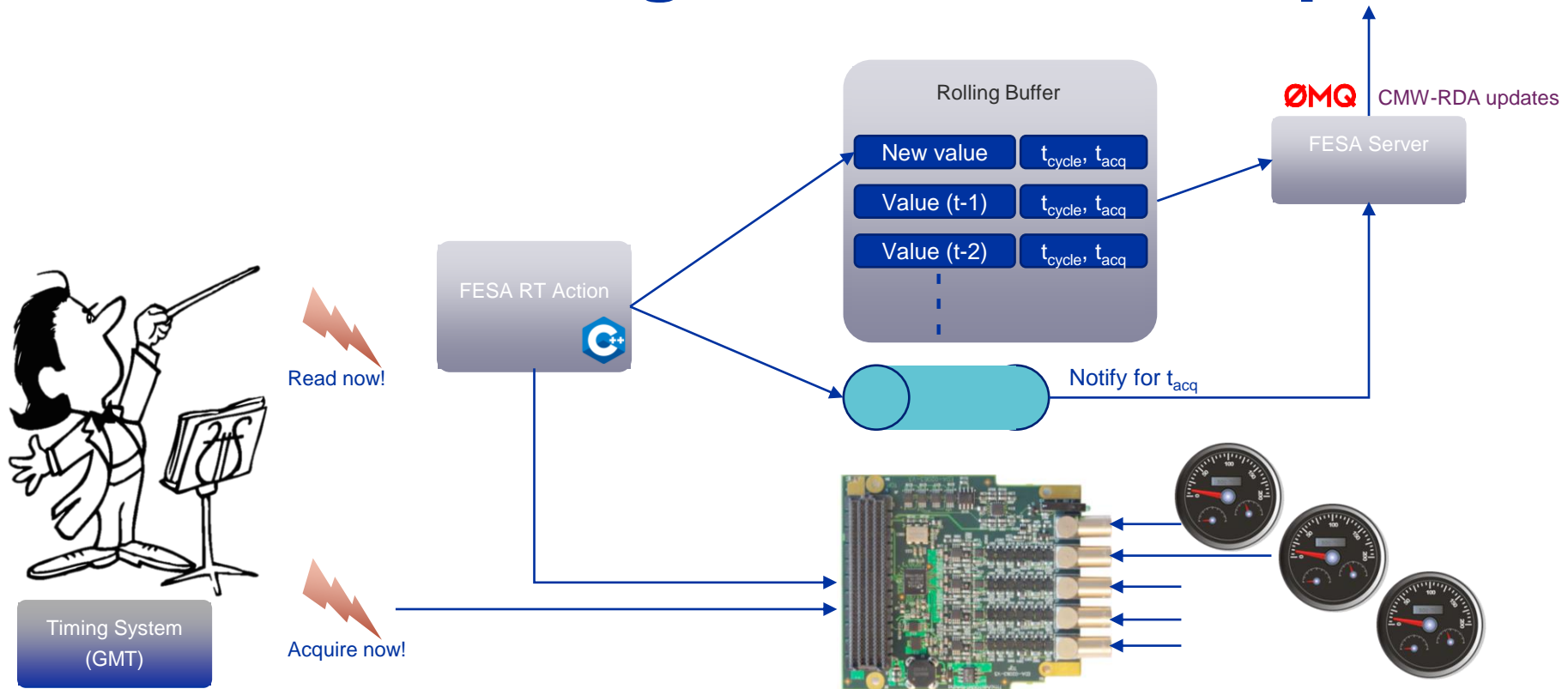
In all the examples, we assume all the configuration is already done and we focus on the run-time aspects

Example 2 – Monitoring

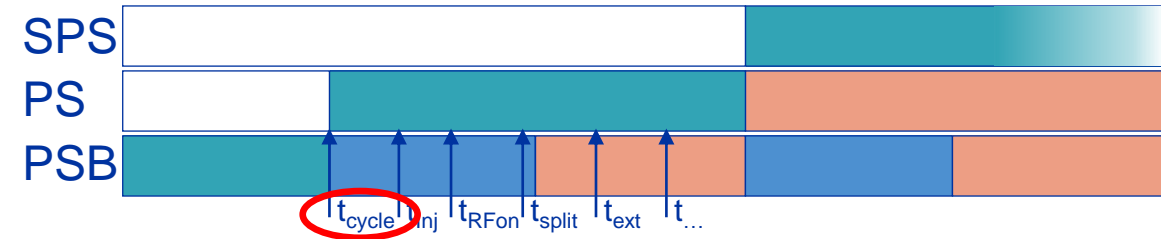
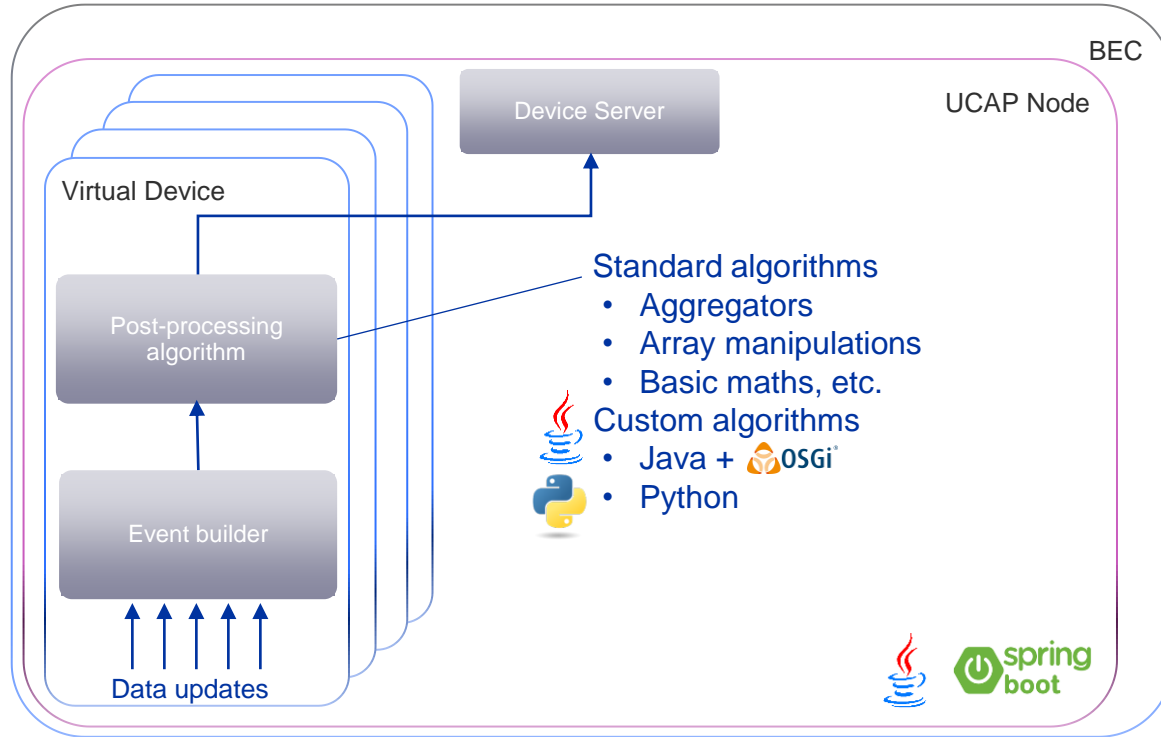


*UCAP: Unified Controls Acquisition & Processing framework

Example 2 – Monitoring – Low-level acquisition



Example 2 – Monitoring – Post-processing



- How to group the incoming data?
→ Start Cycle timestamp (AKA cyclestamp)
- When to trigger the post-processing?
→ Once all the data is there or after a time-out
- How long to wait for late comers?
→ Configurable time-out
- What to do if no data is published?
→ Out-of-the-box monitoring



Example 2 – Monitoring – Diagnostics



Example 2 – Monitoring – Graphical User Interface

CPS:BFAs - CPS.USER.TOP - (INCA)

Controller	UserPermitted	User Permitted To Play	Kicker	Kick Count
BF21P_359_F3_CONTROLLER	false	false	1, 0, 0, 0, 0, 0, 0, 0, ...	1
BF21P_359_F3_CONTROLLER	false	false	1, 0, 0, 0, 0, 0, 0, 0, ...	1
BF21P_359_F3_CONTROLLER	false	false	1, 1, 1, 1, 1, 0, 0, 0, ...	5
BF21P_359_F3_CONTROLLER	false	false	1, 1, 1, 1, 1, 0, 0, 0, ...	5
BF21P_359_F3_CONTROLLER	false	false	1, 1, 1, 1, 1, 0, 0, 0, ...	5

Virtual	Delay[ns]	Delay 21	Strength[V]	Delay to Play	Delay 21 to ...	Strength to ...	Pfn	Agn[V]
PE_BFA21P-V	1000		5000	1000		5000	73	
PE_BFA21-951-V	1000		5000	1000		5000	122	
PE_BFA21-952-V	1000	1000	5000	1000	1000	5000	24	
PE_BFA21-953-V	2000	2000	5000	2000	2000	5000	0	
PE_BFA21-954-V	3100	3100	5000	3100	3100	5000	24	
PE_BFA21-955-V	4200	4200	5000	4200	4200	5000	0	
F16_DFA242-V	1000		11500	1000		11500	0	
F16_DFA2451-V	1030		9000	1030		9000	0	
F16_DFA2452-V	1100		9700	1100		9700	0	
F16_DFA2453-V	2200		10000	2200		10000	0	
F16_DFA2454-V	3100		10500	3100		10500	24	
F16_DFA2455-V	4400		11000	4400		11000	0	

PSState	Mode	Local/Remote	REMOTE	Busy	External Cond.	FAULTY
PE_BFA21P_STATE	OFF	ON	REMOTE	NO		0K
PE_BFA21-095_STATE	ON	ON	REMOTE	NO		0K
PE_BFA21-095_STATE	OFF	ON	REMOTE	NO		0K
F16_DFA242_STATE	ON	ON	REMOTE	NO		0K
F16_DFA242_STATE	ON	ON	REMOTE	NO		0K

LTIM	Event	Start	Delay	Clock Str.	AgnC	AgnCNano
PEX_SBF21P	Disable	PEX.MRF	7620	PAX.TRF	-	-
PEX_SBF21P	Enable	PEX.MRF	7620	PAX.TRF	-	-
PEX_SBF21P	Disable	PEX.MRF	7614	PAX.TRF	-	-
PEX_SBF21P	Enable	PEX.MRF	7614	PAX.TRF	-	-
F16X_SDF242	Disable	PEX.MRF	7652	PAX.TRF	-	-
F16X_SDF242	Enable	PEX.MRF	7652	PAX.TRF	-	-
F16X_SDF242	Disable	PEX.MRF	1600	PAX.TRF	-	-
F16X_SDF242	Enable	PEX.MRF	1600	PAX.TRF	-	-
LTIM	Event	Start	Delay	Clock Str.	AgnC	AgnCNano
PEX_SSNP-BFA	Enable	PEX.MRF	20000	10MHz	728	728000600

Timing App Suite

General Cycles Rules

Filter LEI cycles

Filter PSB cycles

Filter CPS cycles

Filter SPS cycles

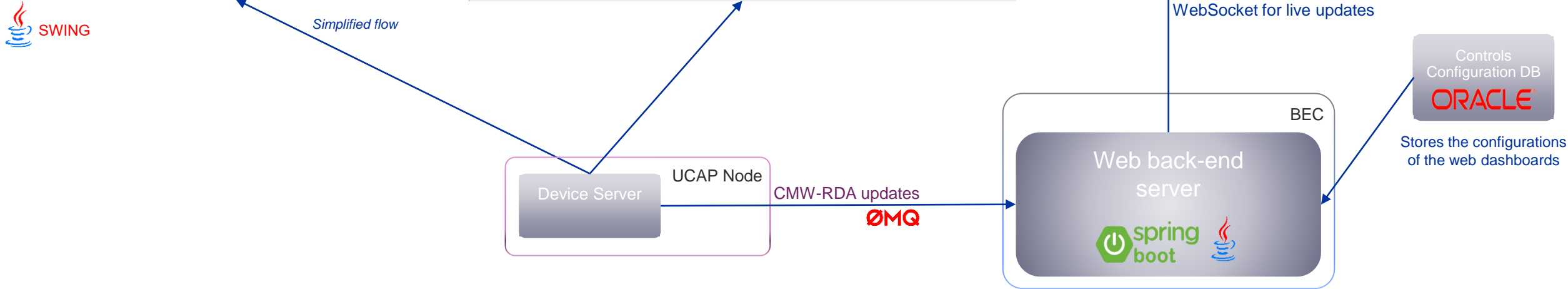
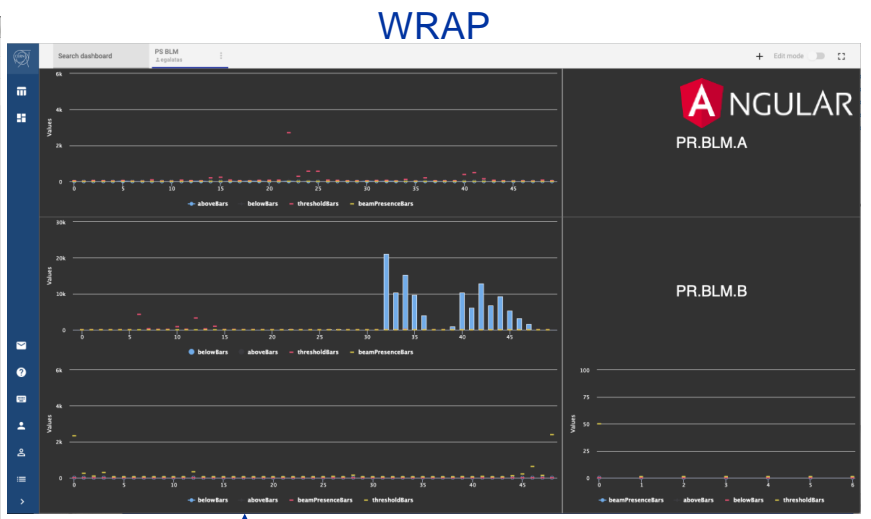
Basic Periods: 6

MD_26_L7200_Q20_North_Extraction_2021_V1

LHC#1b_INDIV_DispSPS

Name: LHC#1b_INDIV_DispSPS
Length: 2 BP
Desc: for dispersion measurement in SPS

MISC: Destination 16: The indication for fast, slow or MT extraction is not set (MISC)



Use Case 3: Logging

- **Who:** Accelerator physicist
- **What:** Store acquisition values of many control devices long-term and perform analysis
- **Involved controls components:**
 - Timing (Synchronisation)
 - FESA (Real-time hardware control)
 - CMW (Controls Middleware)
 - NXCALS (Data Logging)
 - SWAN (Web-based analysis tool)

In all the examples, we assume all the configuration is already done and we focus on the run-time aspects

Example 3 – Logging



```

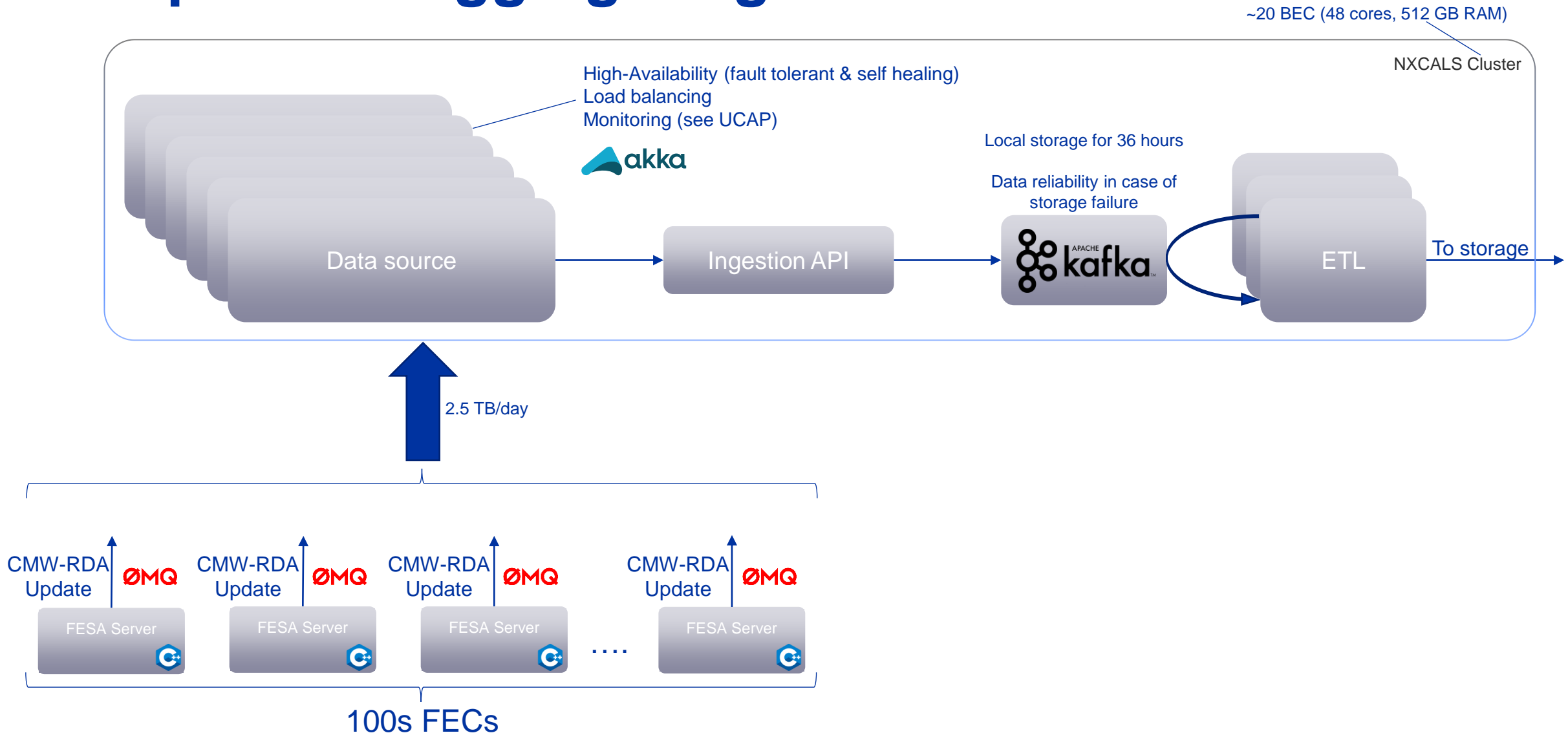
In [1]: from cern.ncxals.epi.extraction_data.builders import *

In [2]:
def getQuery():
    builder (spark).builder (builder).system ("CW")
    .start ("2018-11-15 00:00:00")
    .end ("2018-11-15 00:10:00")
    .variables ("SW", "CT", "AQ")
    builder (builder)
    da = da.withColumn ("val", da["sws_val.elements"])
    pandas = da.select ("sws_val.timestamp", "val").sort ("sws_val.timestamp").sample (0.1).toPandas()

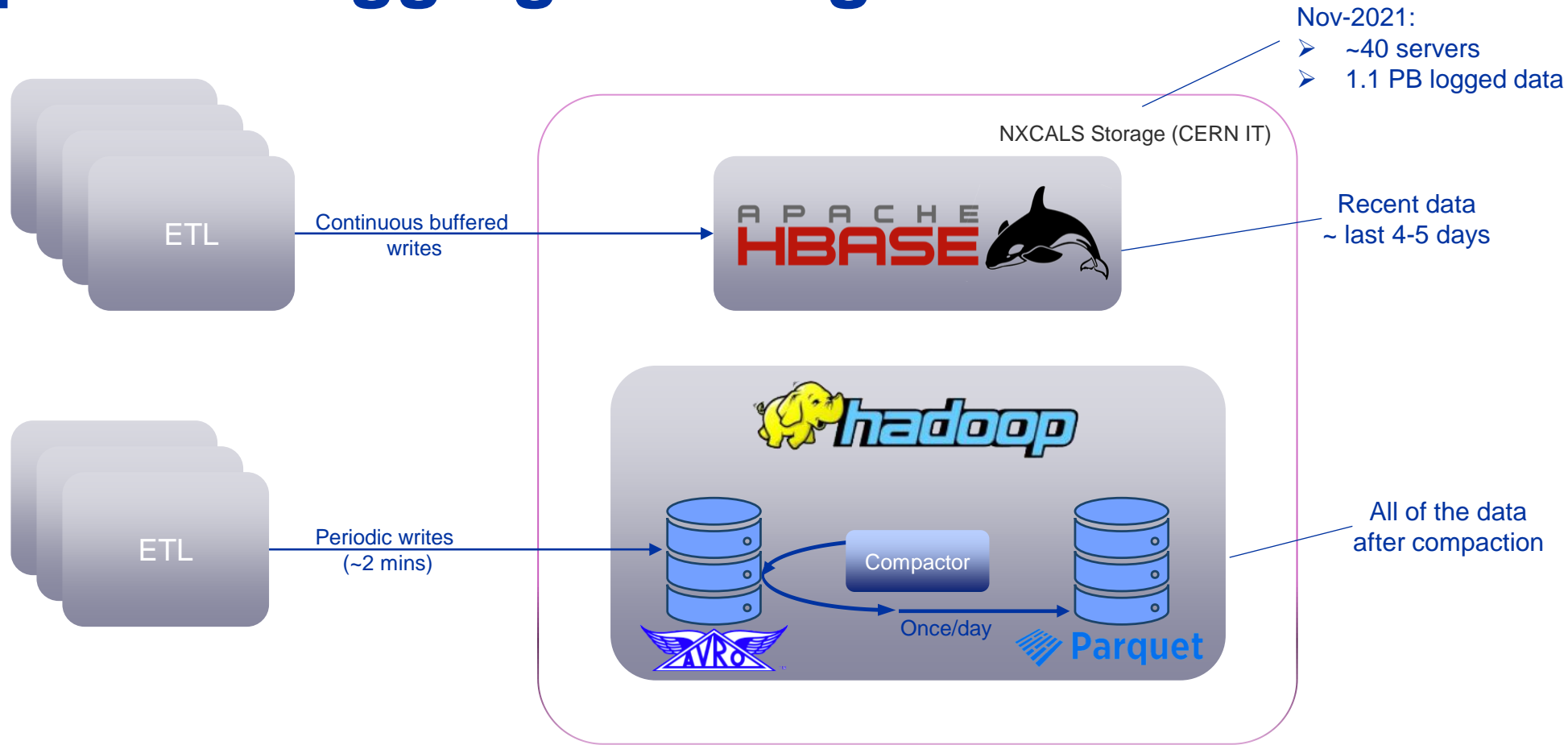
In [3]:
%matplotlib notebook
import matplotlib.pyplot as plt
t1 = time.time()
t2 = t1 + 10
plt.plot (t1, y1)
    
```

*SWAN: Service for Web-based ANalysis

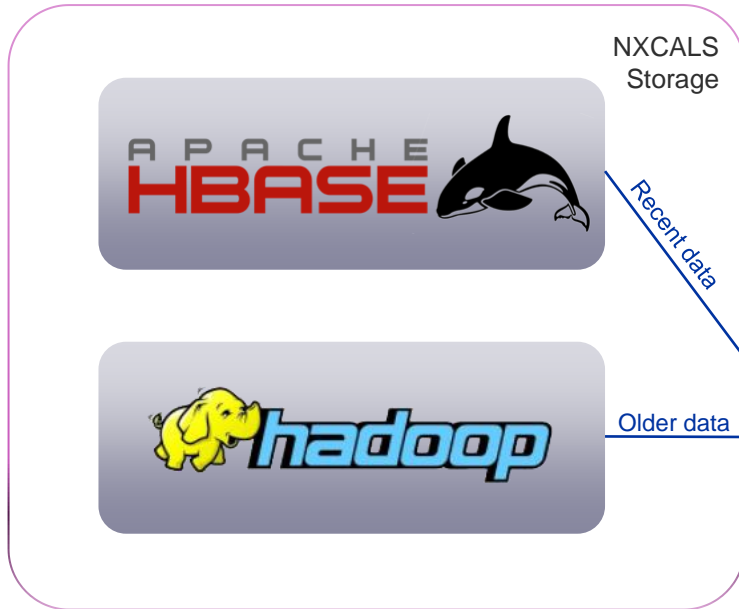
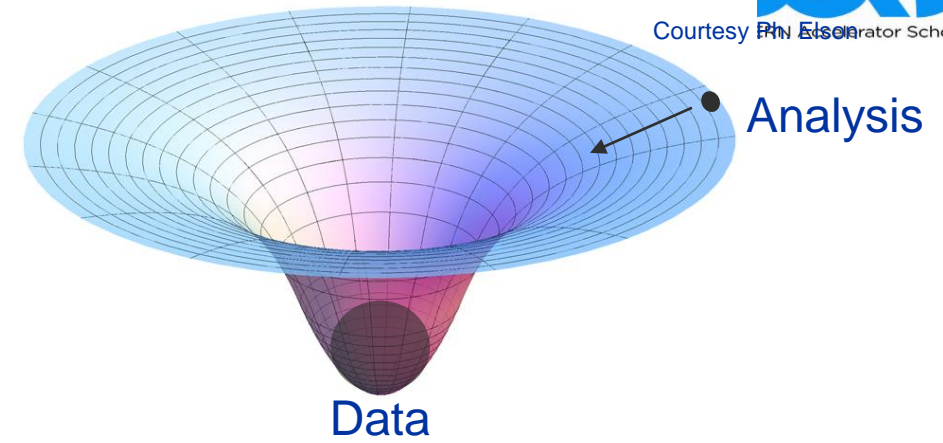
Example 3 – Logging - Ingestion



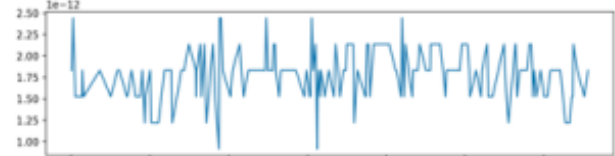
Example 3 – Logging - Storage



Example 3 – Logging - extraction



```
In [1]:  
from cern.nxcale.api.extraction.data.builders import *  
  
In [2]:  
dq=DataQuery\  
    .builder(spark).byVariable().system("CMW")\  
    .startTime("2019-11-15 00:00:00")\  
    .endTime("2019-11-15 00:10:59.999")\  
    .variable("DMS1.PC10-AQM").buildDataset()  
ds = dq.withColumn("val", dq["nxcale_value.elements"]())  
pandas = ds.select("nxcale_timestamp", "val").sort("nxcale_timestamp").sample(0.1).toPandas()  
  
In [3]:  
%matplotlib notebook  
import matplotlib.pyplot as plt  
t1,v1=pandas.values.T  
t1/=1e7 # to seconds  
plt.clf()  
p1.plot(t1,v1)
```



Want to know more?



CERN Beams Department (<https://beams.cern/>)

Introduction to BE-CO Control System, 2019 Edition,
S. Deghaye & E. Fortescue, CERN, 2020.
(<https://cds.cern.ch/record/2748122>)



Tango Controls (<https://www.tango-controls.org/>)



EPICS (<https://epics-controls.org/>)





home.cern