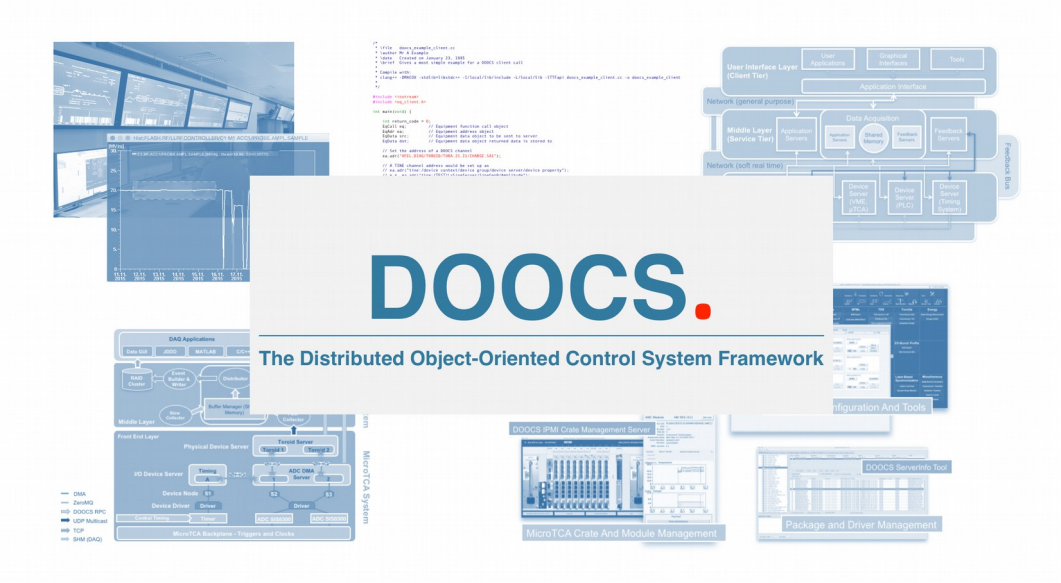
A Tour de DAQ

The DOOCS-based Accelerator Data Acquisition System – A Brief Overview



Based on DOOCS Lecture And Tutorial Series

Tim Wilksen, MCS
Vladimir Rybnikov, FTX
Hamburg, 26.10.2022

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DESY.

Outline

Outline

- Motivation
- Essentials
- Basics
- Data Tools
- Status
- Outlook
- References

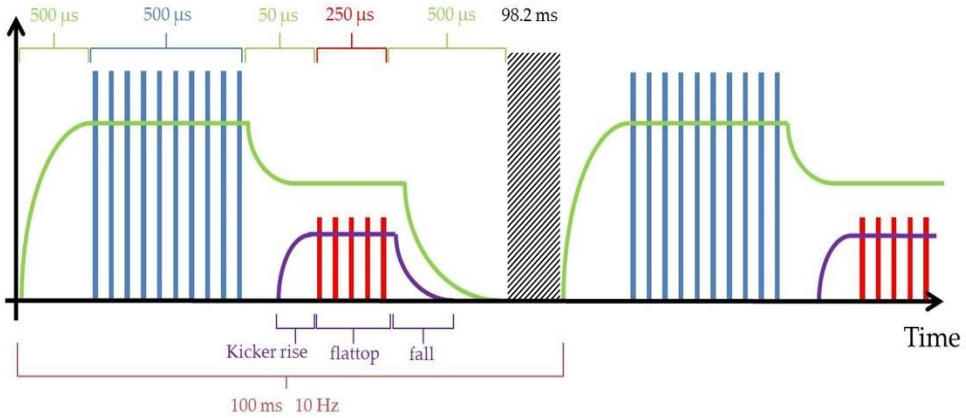
Motivation

Motivation

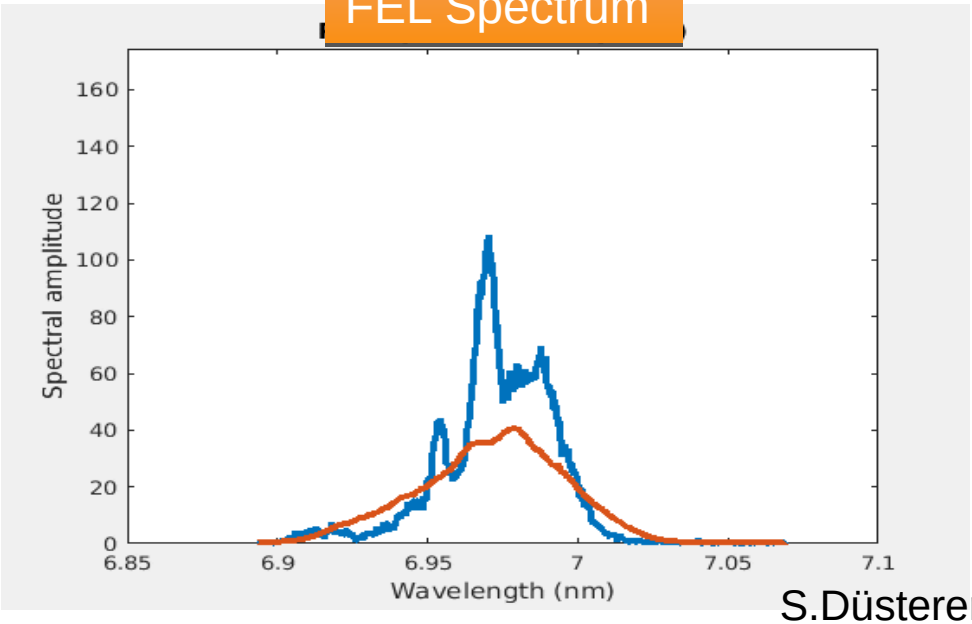
DESY FEL are “pulsed machines”

- EuXFEL, FLASH et al are FEL accelerators operated at 1 - 10 Hz shot repetition rate with several hundred up to thousands of bunches per shot
- Shot- or train-based data with inner structure - bunch pattern with up to 4.5 Mhz repetition rate
- Conditions and properties may vary from shot to shot and especially different for each individual bunch in one train (length, energy, arrival time, …) of electron bunches and photon pulses

Interest in shot-synchronized and pulse-resolved data analysis capabilities!



FEL Spectrum



S.Düsterer

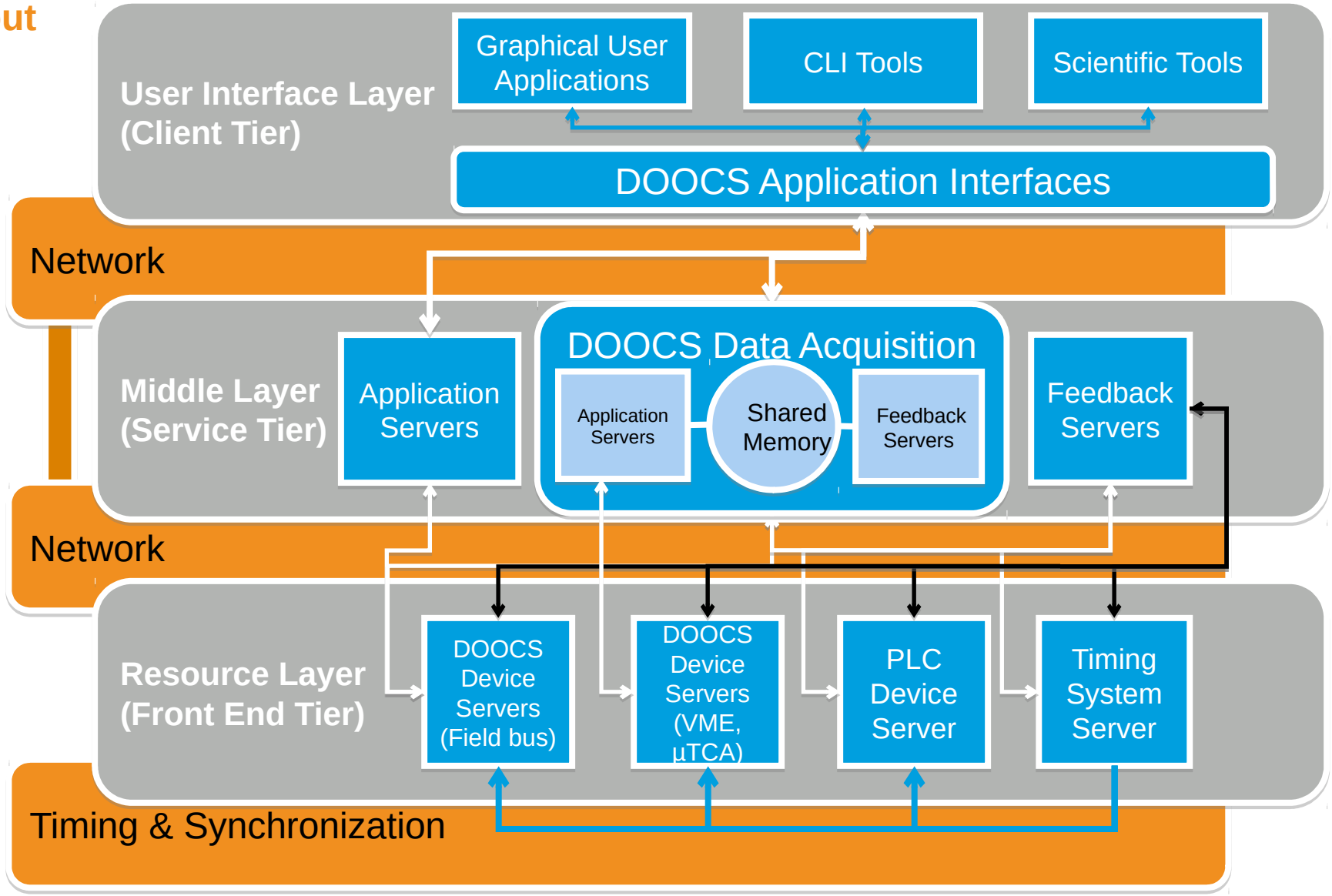
Motivation

Basic Approach of an Accelerator DAQ

- **Collect and store** all beam relevant data of the linac.
- Acquisition has **access to all data** – accelerator diagnostics and user experiments (when possible).
- **Synchronize all data** and make it available for middle layer services and user applications.
- Integrate **accelerator model** and **simulation** into DAQ (Virtual accelerator or digital twin)
- **Integrate** data acquisition into control system concept, not just as a separate entity for subsystems (e.g. beam lines, diagnostics, cameras).
- Provide the **tools to analyze** the stored data for loal and remote users
- Improve, better understand and maintain the linac
  - Error statistics: find reasons of faults, improve reliability
  - Operation optimization: find best parameters
  - Allow experiments to correlate their measurement with the machine parameters

Motivation

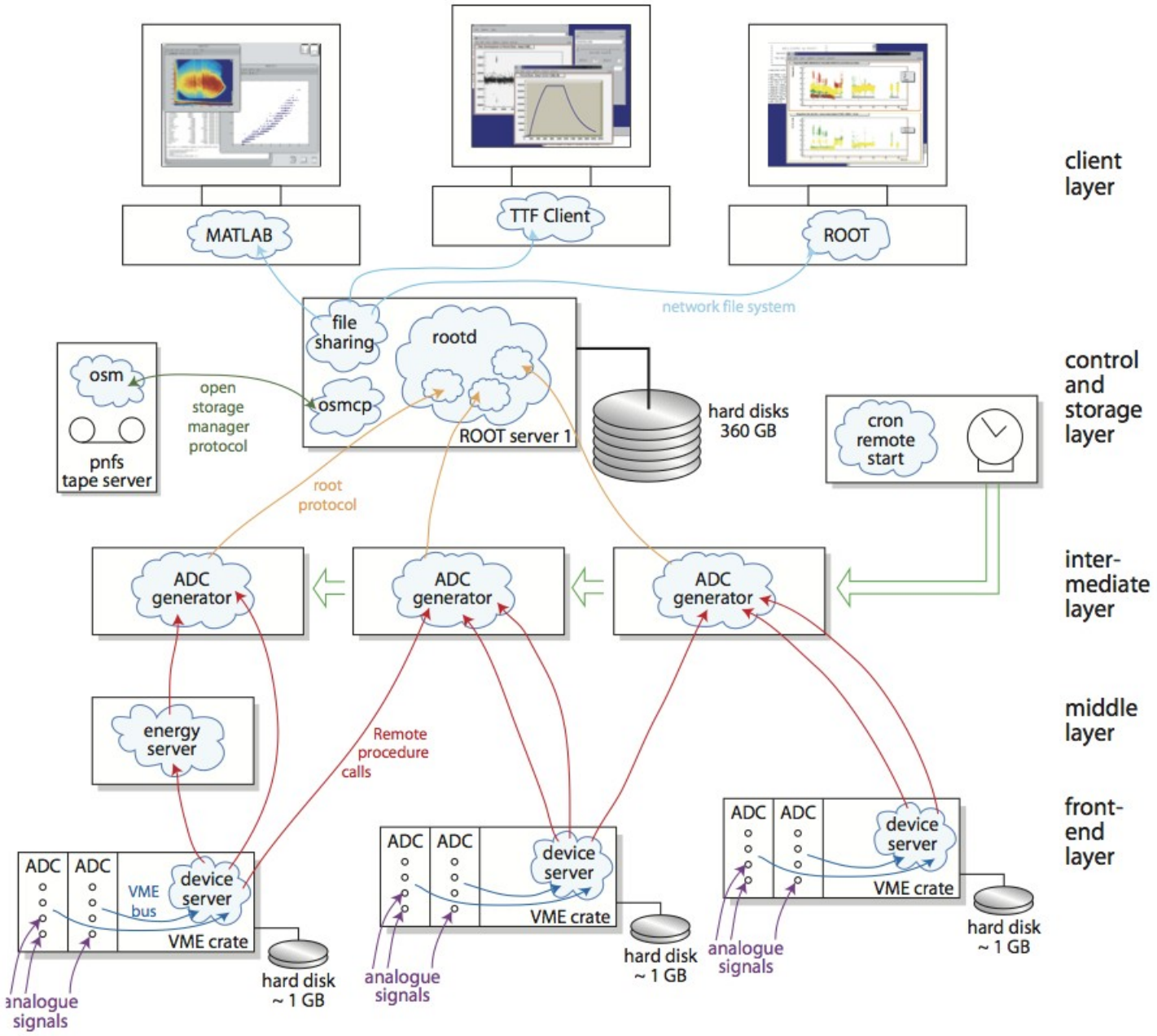
A DOOCS Layout



Motivation

A bit of history … TTF2

- Idea P. Schmüser et al at TTF(2) ~ 2001
- Shot-synchronized data recording for bunch-related data.
- Based on ROOT and DOOCS (written by C. Lackas and R. Ischebeck).
- ADC generator farm (collector)
- Nice concept and feasibility study, some performance issues and cumbersome data retrieval.
- Redesign for FLASH started 2003.
- **DAQ Collaboration (2003)**
- Use concepts found in HEP-style DAQ systems runs, central configuration, event idea, file storage.
- **First working DAQ instance when FLASH started for real (2005).**

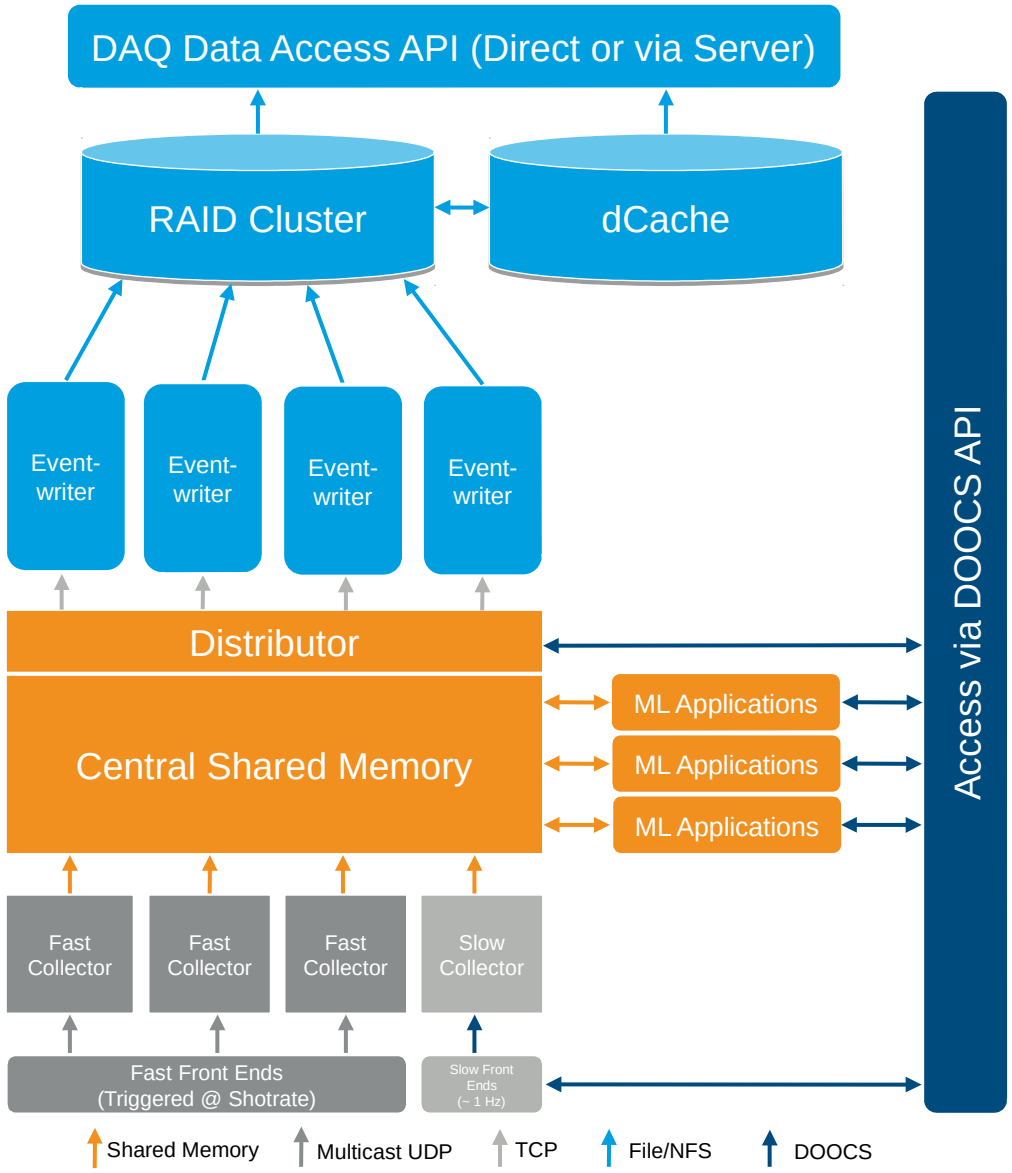


Essentials

Essentials

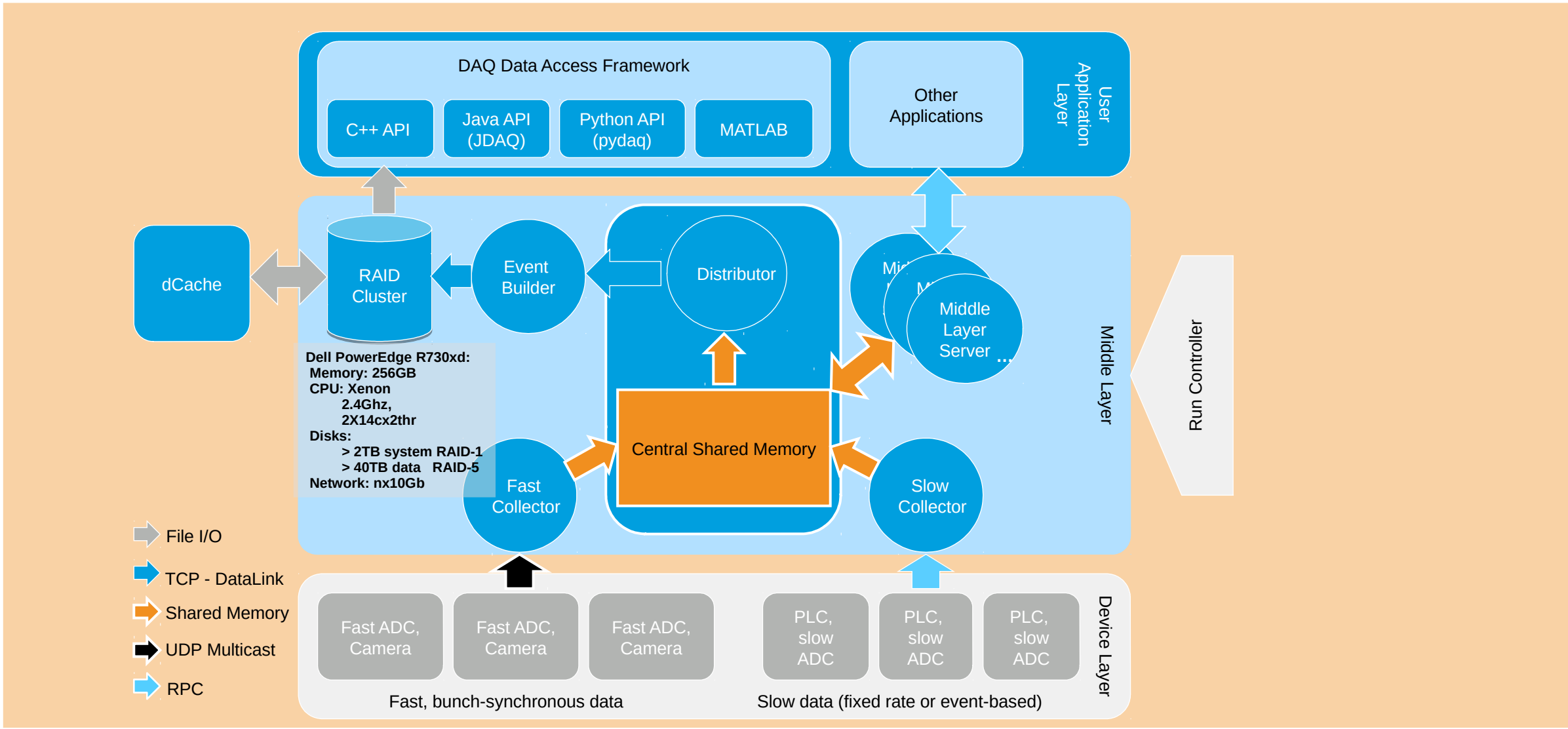
Design Specifics

- Derived conceptually from a high-energy physics experiment DAQ
- Data has event-like structure recorded at shot repetition rate (10 Hz)
- Event **record holds all train related and bunch-resolved data** as configured for a given time period (run-based concept)
- **Collection** of train-related data **via UDP multicast** w/ shot/train/event ID synchronization
- Central shared memory for assembling event record and enabling fast online analysis
- **Middle layer** applications perform **online processing** – results can be written back to current event record in shared memory
- Simple RAID cluster for intermediate storage w/ proprietary file format
- C++-, Java-, MATLAB-, (Python-) API for batch processing and data extraction
- Java/Python/MATLAB tools for offline data retrieval: file manager, data browser, data extraction, taping, conversion
- Run-based **configuration** and **bookkeeping**



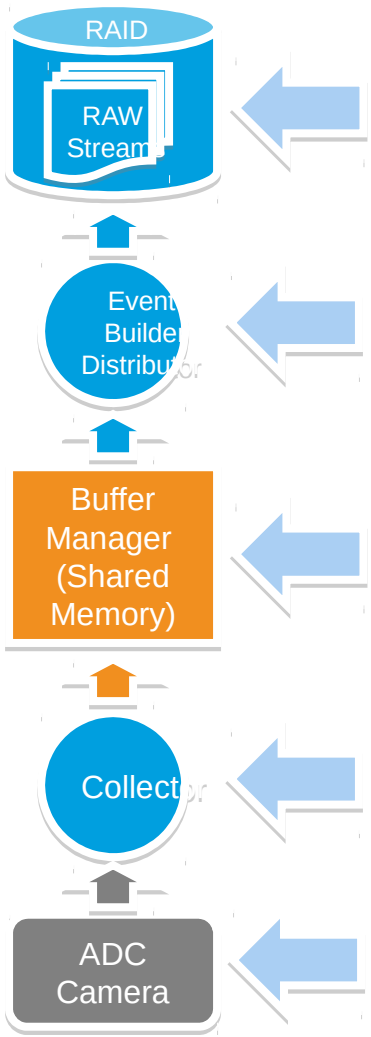
Essentials

Layout



Basics: Configuration

Basics - Run Control and Configuration



DAQ Configuration by Run Controller

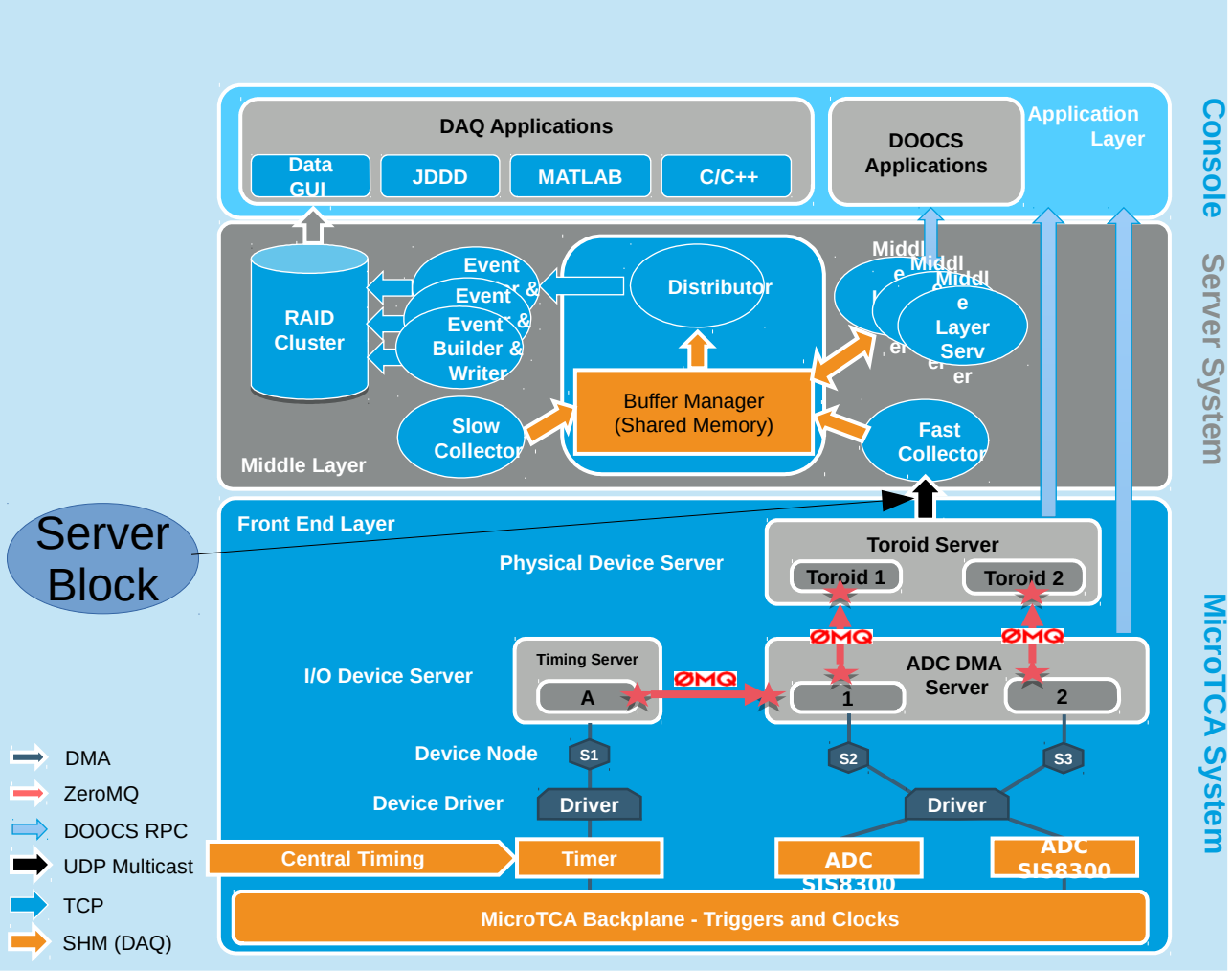
- Configuration properties i.e.
  - Run mode – defines e.g. sample size and systems included.
  - Stream selection – which data goes into what raw files
  - Subsystem and channel/server block selection.
- Hosted on DESY ORACLE DB system.
- Java and Python tools to configure run control DB.
- Run controller sends XML configuration files to DOOCS servers. Completely automated start up and configuration of data acquisition.
- Finite state machine for run controls, HEP-style with accelerator add-ons.

Basics: Data Collection

Basics

Data Collection – Timing System and trainID / Shot Number / Macropulse Number / Event ID

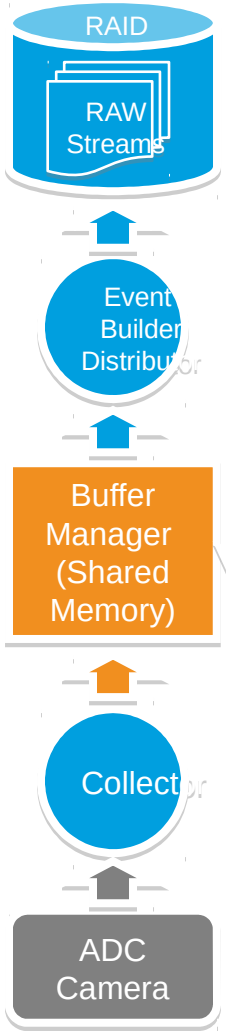
- Common timing system hardware (x2timer – MTCA.4 AMC) in every front-end connected via fiber optics net to master.
- **Unique identification of individual shot data** by sending timing system information i.e. clock signals, trigger events, pulse number etc. to all front-end devices.
- A timer server program on every MicroTCA system provides the timing system information also via ZeroMQ to applications.
- The **stamped data is sent via a push-type protocol based on UDP multicast to all subscribers** e.g. also DAQ instances.



Basics: Data Assembly

Basics

Data Assembly



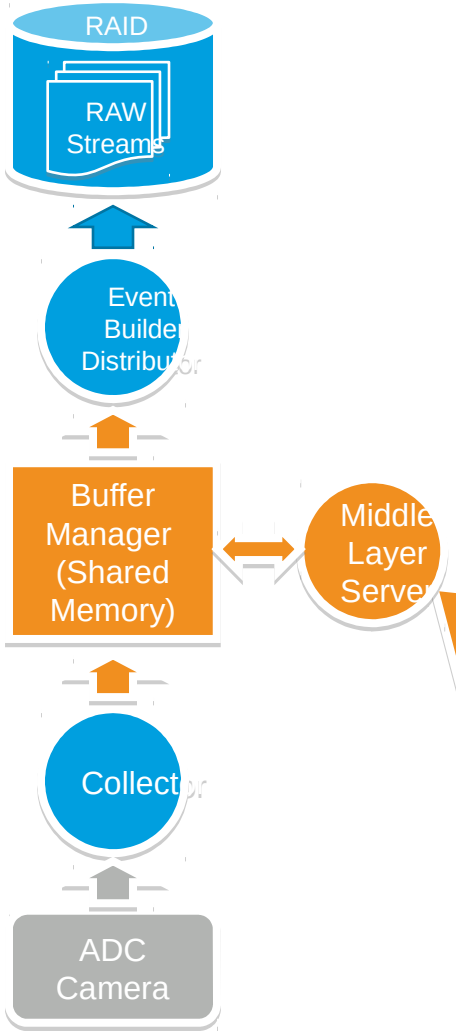
Buffer Manager – Central Shared Memory

- Shared memory area holds several, complete event records – as long as required by shared memory clients.
- Buffer manager (BM) is **main instance** of controlling shared memory.
- A collector process writes server blocks to shared memory, hence it is a **producer**.
- Distributor reads all server blocks belonging to one event ID (shot) from shared memory and distributes the event record, once it is complete, to the event writer processes. The distributor also happens to be the BM **master**.
- Other clients can be middle layer server, reading events from shared memory (**consumers**) and even writing new server block back to the shared memory as a result from some calculation or processing in general. I.e. they can be consumers and producers.

Basics: Data Processing

Basics

Data Processing – DAQ Middle Layer Server



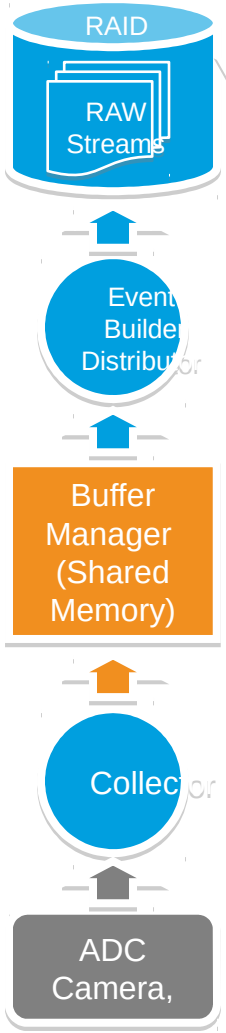
Middle Layer Servers or Application Servers

- ***Middle layer server is a client*** with respect to the buffer manager. Hence it has access to all announced event types and shot-synchronized data..
- Along with reading data (consumer) can also process it and write results back (producer) with a new event type being part of the same shot data.
- Notified when requested data (subscribed server blocks) are available in shared memory.
- Can perform many kinds of computational tasks.
- Uses Python, MATLAB or C/C++ for calculations, therefore standard mathematical libraries are vastly available.
- Sufficient response time for many operational tasks and some automation purposes. Examples: Energy measurement, photon wavelength and energy measurement, gas monitor detector server, LLRF server, orbit server, quench detection server,

Basics: Data Storage

Basics

Data Storage



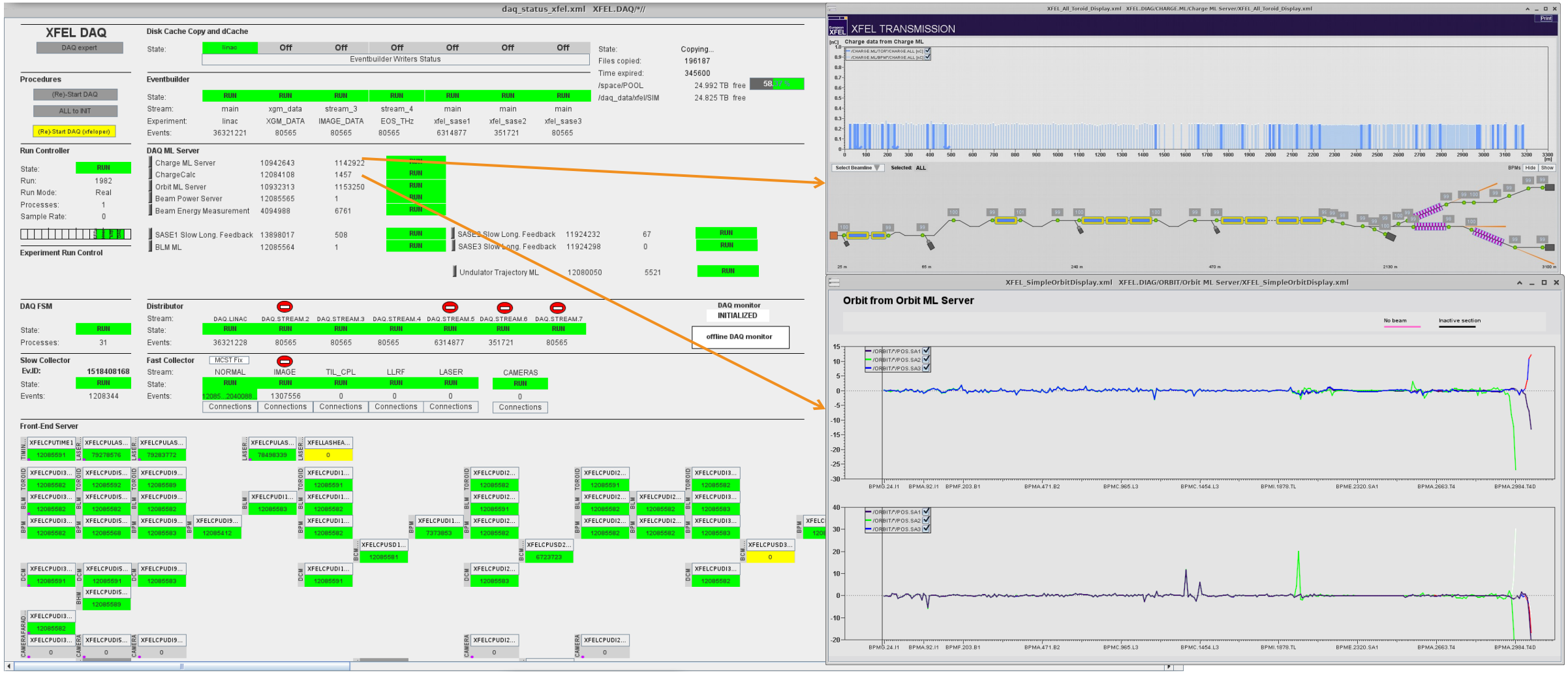
Data Storage

- Dedicated storage nodes with disk space for accelerator and user experiments.
- Data is stored in raw file format, similar to a column-wise n-tuple (HEP data analysis format) - a custom raw format to increase performance.
- Selected data is transferred to the photon science dCache system – stage and tape system transparent for the end-user (DESY/FNAL development, GRID enabled).
- All data from user experiments is transferred to dCache for long-term archival.

Applications

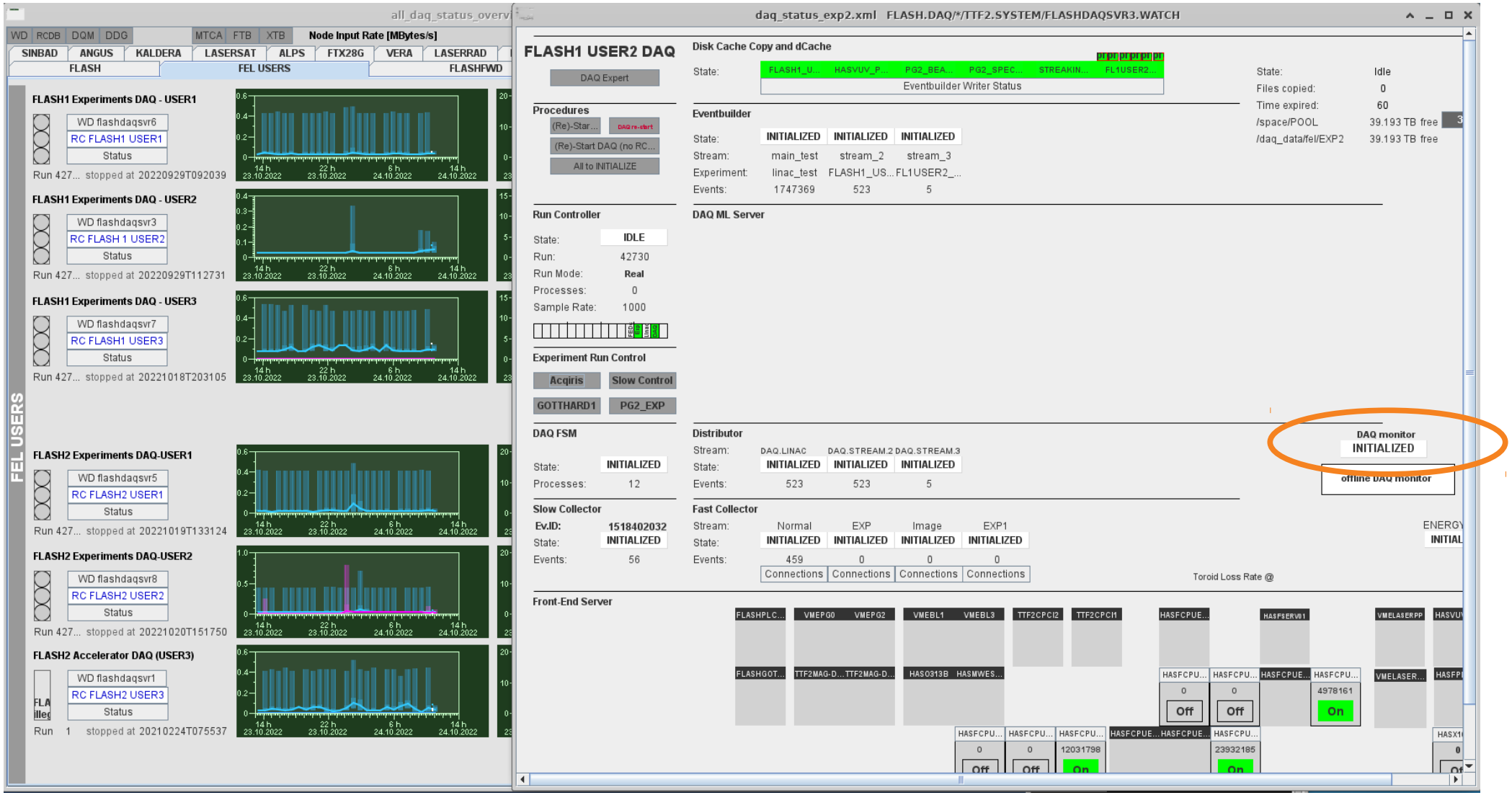
Applications

FLASH/XFEL DAQ Instance – monitoring and feed-back servers



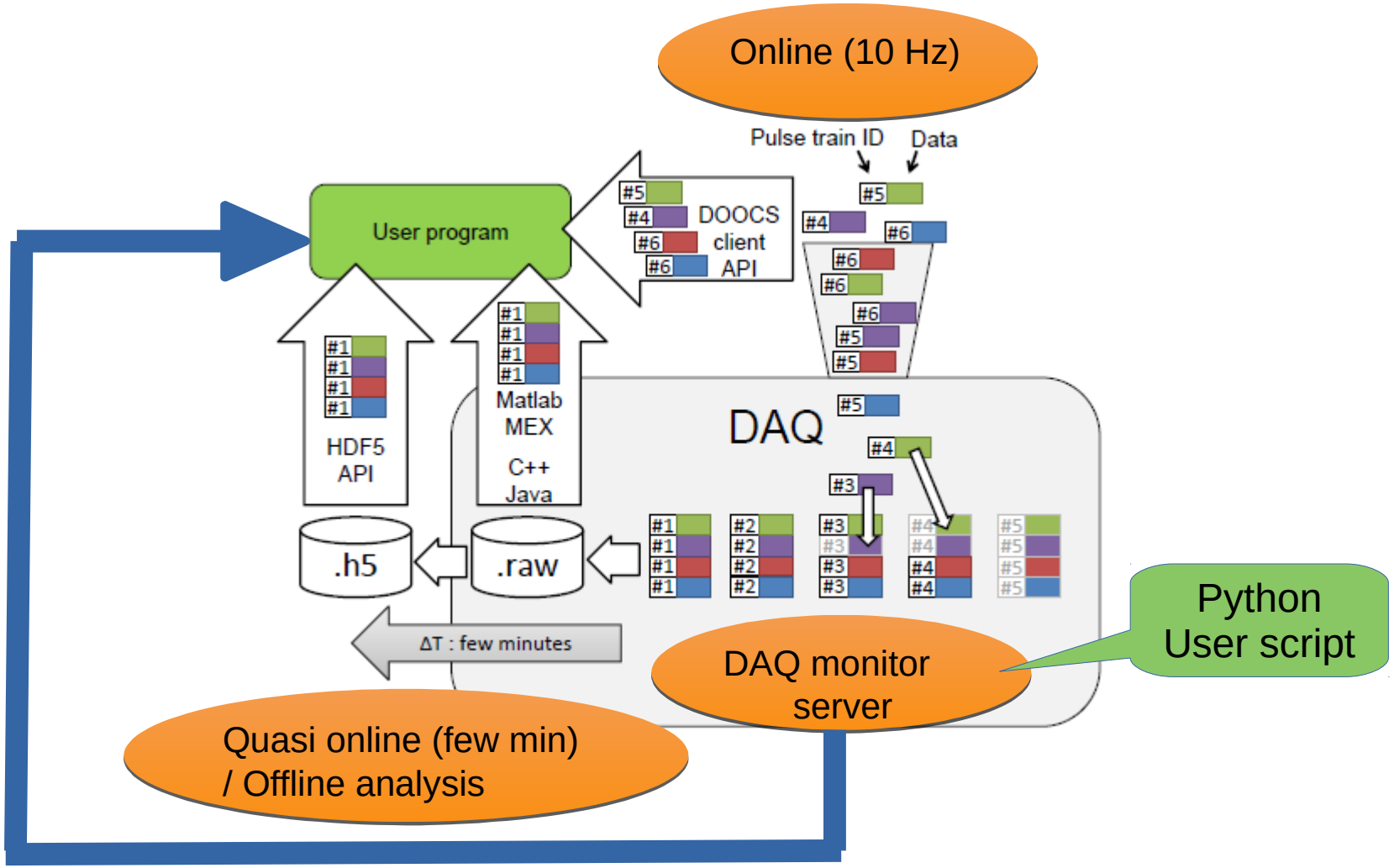
Applications

Experiment Data Acquisition – Photon Science



Applications

Experiment Data Acquisition – Photon Science

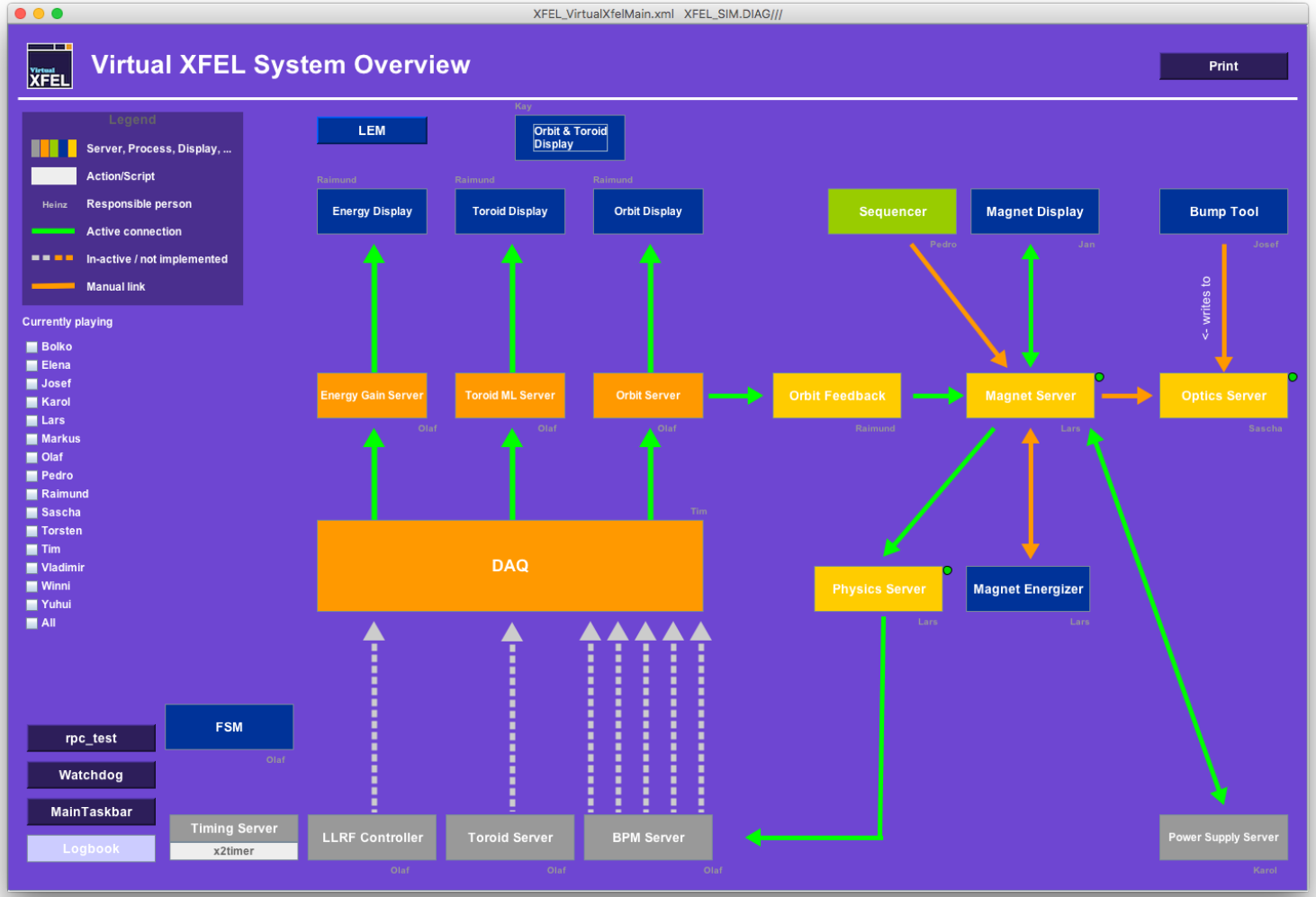


Experiments need pulse train synchronous data

Applications

Virtual DAQ Instance – Virtual XFEL

- Simulated front end data fed into the same software as for the “real” DAQ instances
- Fully working system up to high-level controls w/ limited physics models
- Test-bed for testing new software and concepts for high-level applications
- Independent of real accelerator control system, open for developers and testers anytime



Data Tools

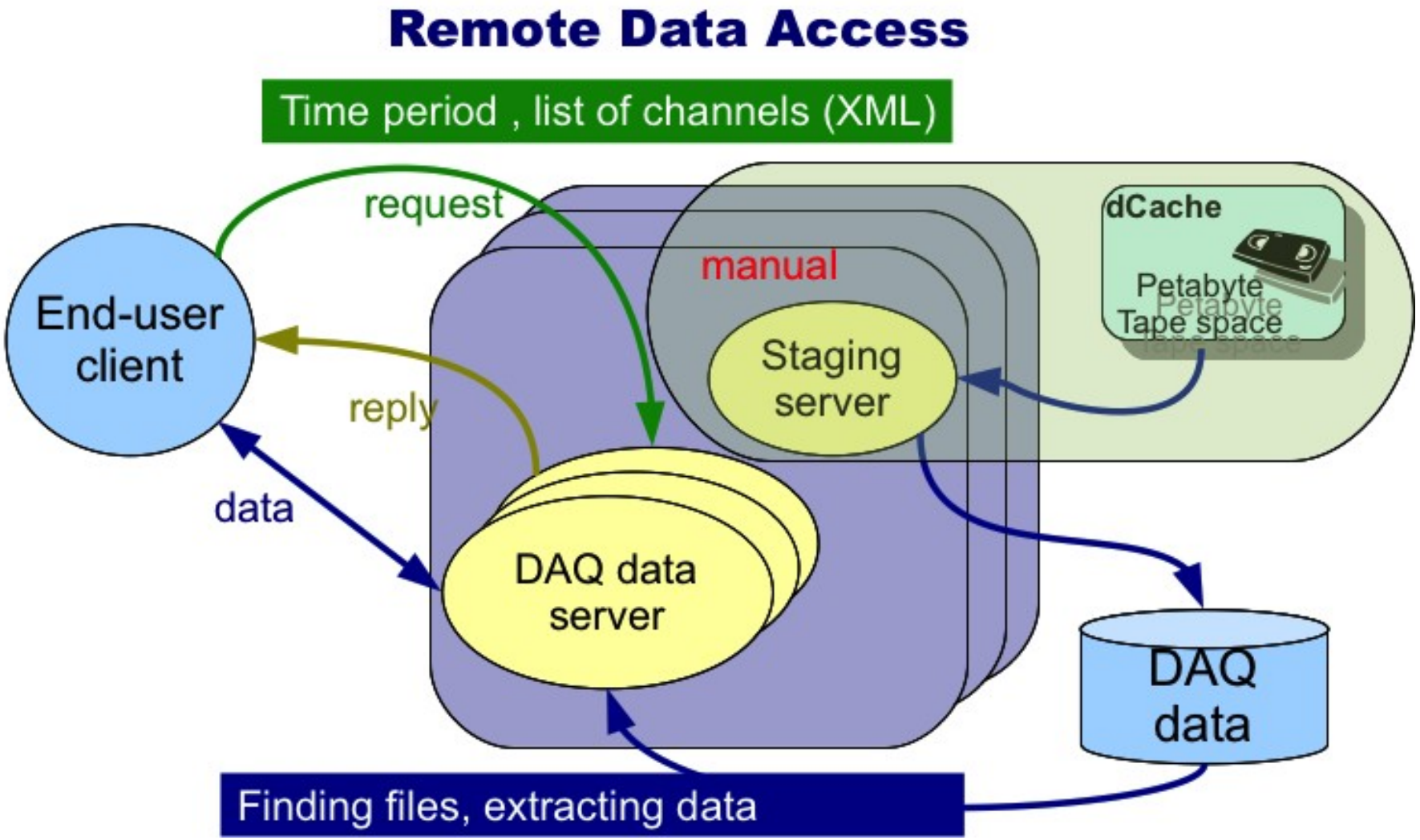
Data Tools

DAQ Data access types

Type	Data Source	Purpose	Options
On-line	Buffer Manager	Monitoring Data Processing Feed-backs	Writing results to BM
Off-line	Data Files	Data analysis	Local, Remote

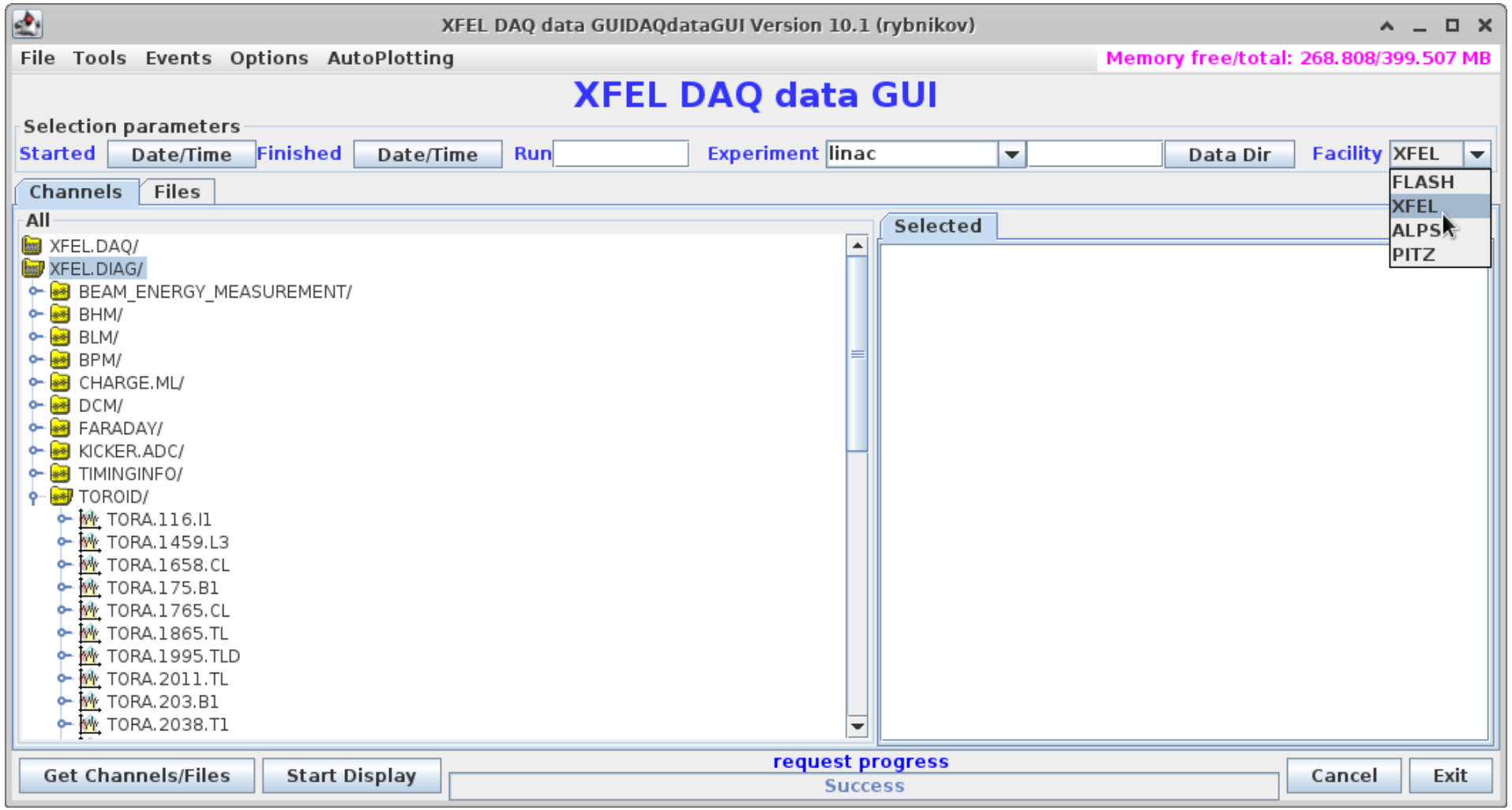
Data Tools

DAQ Data (Provisioning) Server (Farm)



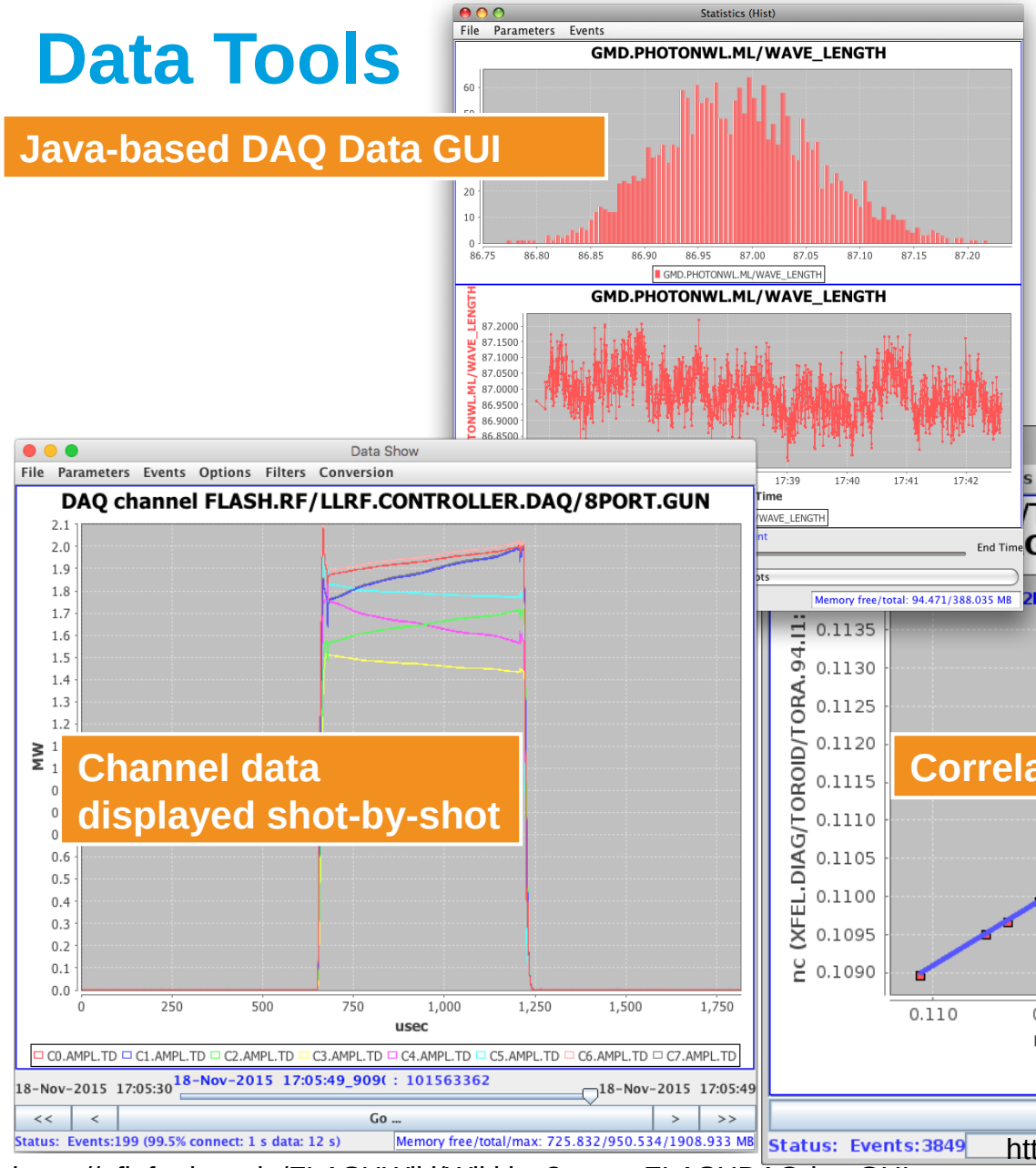
Data Tools

DAQ Data GUI



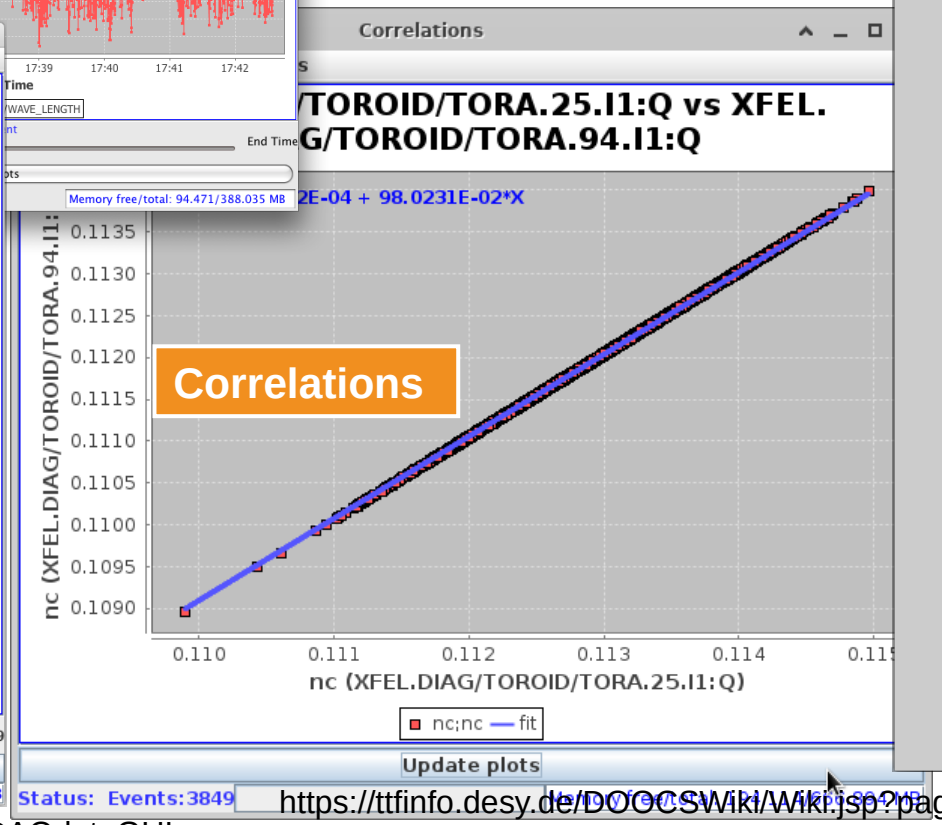
Data Tools

Java-based DAQ Data GUI



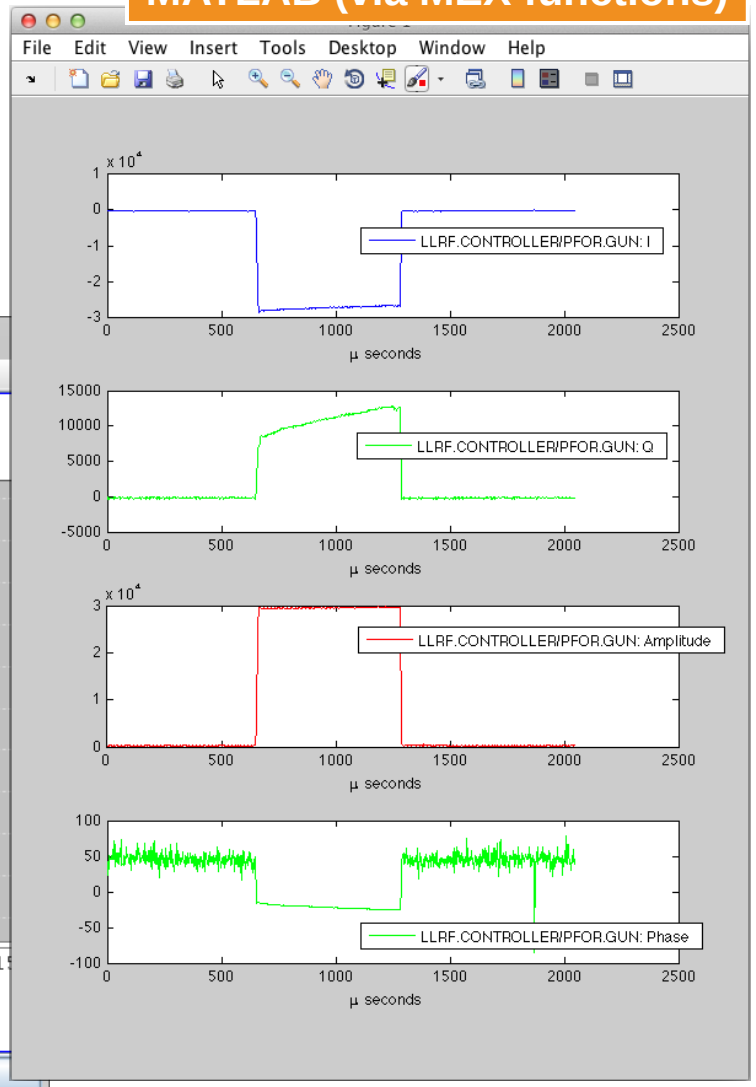
Channel data displayed shot-by-shot

https://ttfinfo.desy.de/FLASHWiki/Wiki.jsp?page=FLASHDAQdataGUI



Correlations

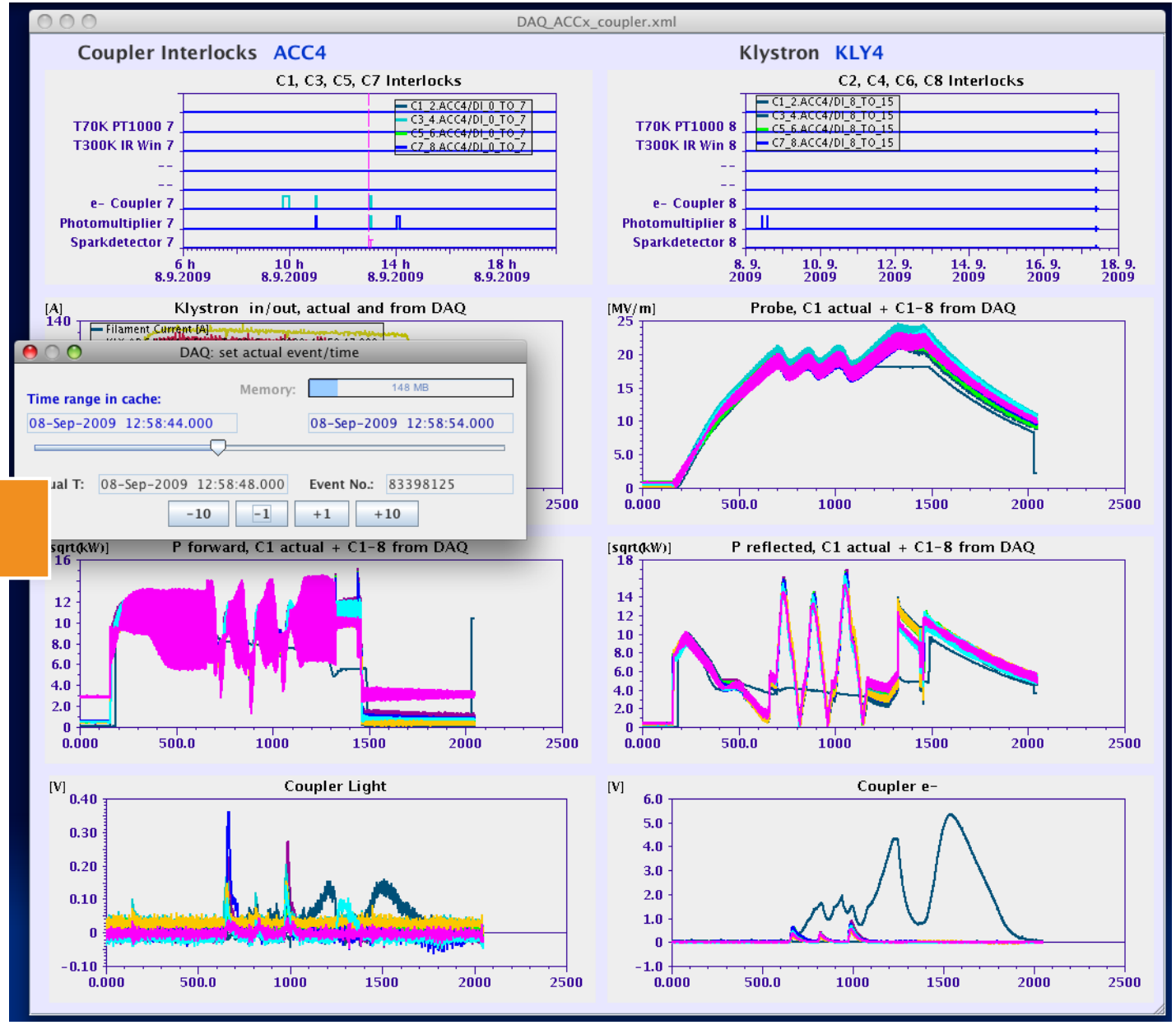
MATLAB (via MEX functions)



https://ttfinfo.desy.de/DOOCSWiki/Wiki.jsp?page=MATLABClientInterface

Data Tools

JDDD Based DAQ Data Viewer



JDDD-based Panel using DAQ Data Server

Data Tools

Python-base DAQ Interface

https://confluence.desy.de/display/DOOCS/Python+DAQ+Client+Interface

Confluence Spaces Create Search Log in

DOOCS DOOCS

PAGE TREE

- Documentation
  - DOOCS Release
  - DOOCS Installation Manual
  - DOOCS Developer Manual
    - C++ Client Interface
    - Java Client Interface
    - Python Client Interface
    - **Python DAQ Client Interf**
    - MATLAB Client Interface
    - C++ Server Interface
    - C++ Server Interface - Arch
    - C++ Server Interface - D_fc
    - C++ Server Interface - D_fc
    - C++ Server Interface - DAQ
    - Driver Development
    - Code Development
    - Code Management
    - Libraries Reference
  - DOOCS Application Manual

Space tools

Pages / ... / DOOCS Developer Manual

Python DAQ Client Interface

Tim Wilksen posted on 04. Jan. 2022 13:00h - last edited by Vladimir Rybnikov on 12. Oct. 2022 08:34h

Python3 FLASH/XFEL DAQ data access interface (pydaq)

Python3 bindings to the FLASH/XFEL DAQ client API (C/C++). These bindings allow to extract DAQ data as a list of dictionaries containing the data and meta information for requested DAQ channels. The data in a dictionary belongs to one DAQ channel. The data in the list is synchronized with the same Event ID (macropulse) in an accelerator.

About pydaq

The Python3-DAQ binding **pydaq** uses the standard Python-C/C++ API as well as a number of DAQ libraries (DataCache, DAQsvrutil, TTF2XML, TTF2evutl, DAQReader, lzo2) which are linked dynamically when pydaq gets imported. In addition the pydoocs module and the DOOCS client API libraries are required.

Environment setup

In order to use pydaq, the shell environment variables have to be set up properly. This has to be done in the same way as in the corresponding instructions for Python Client Interface.

There are two additional environment variables required to set the default DAQ data search path (TF2_DAQ_DATA_PATH_LIST, DESY_PNFS_PREFIX). They can be set by the command

```
source /home/doocsadm/bm/python/DAQ/setup.sh
```

The result depends on the host name. All XFEL DAQ data processing should be done on computers with names starting with "xfel" (xfeluser1, etc).

Provided Functions

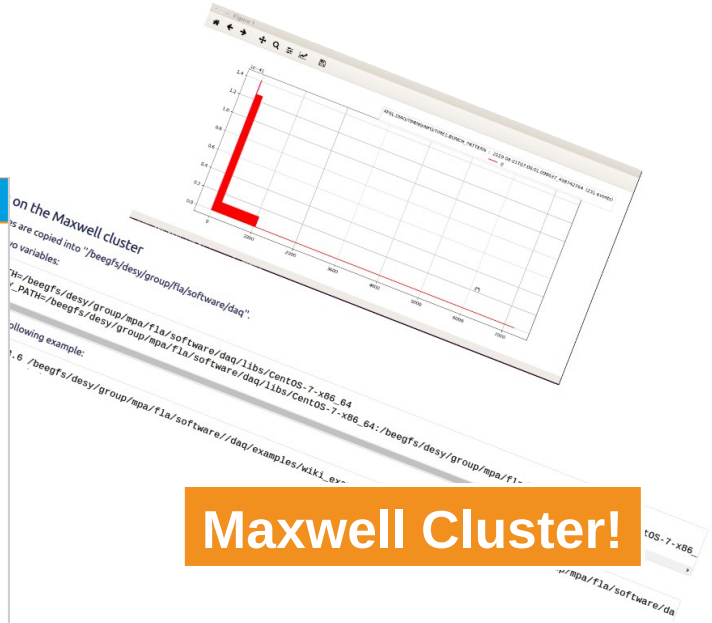
The pydaq module contains 3 functions:

`connect`, `getdata` and `disconnect`.

The short descriptions of the functions as well as simple examples are given below. A more comprehensive example can be found below.

connect

The `connect` function builds up a request for the DAQ data for selected DAQ channels or a list of DAQ channels with some statistics. If a list of channels is requested (and optionally list of files) `connect` will return it if no failure occurs. In case of a request for DAQ data an empty list is returned on success. In case of any problem a `PyDaqException` will be thrown. The available parameters for the `connect` function can be printed by calling the function without any argument like the following.



Data Tools

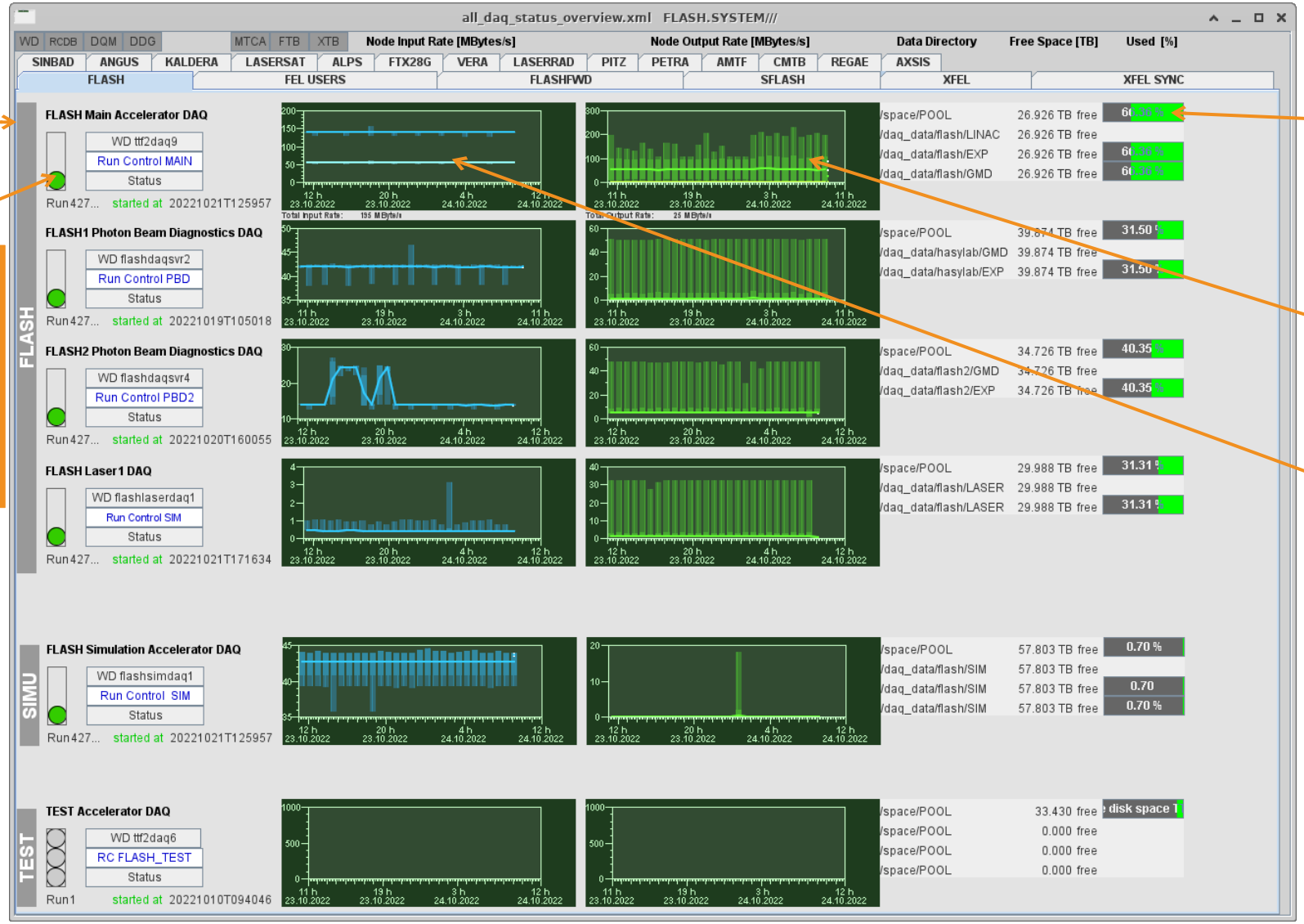
FLASH

DAQ Instance Monitoring

Instance:
MAIN DAQ

Traffic Lights == Health Status:

Grey = Off/Idle
Green = OK
Yellow = Still OK
Red = Take Action, check Status Panel



Disk Space:
green or yellow = OK

Output Rates:
OK if > 0

Input Rates:
OK if > 0,
If dip occurs -> FED problem

12 + 2 DAQ instances at FLASH:
FLASH MAIN DAQ, User DAQs, TEST DAQ, …

Data Tools

FLASH MAIN DAQ Status Panel

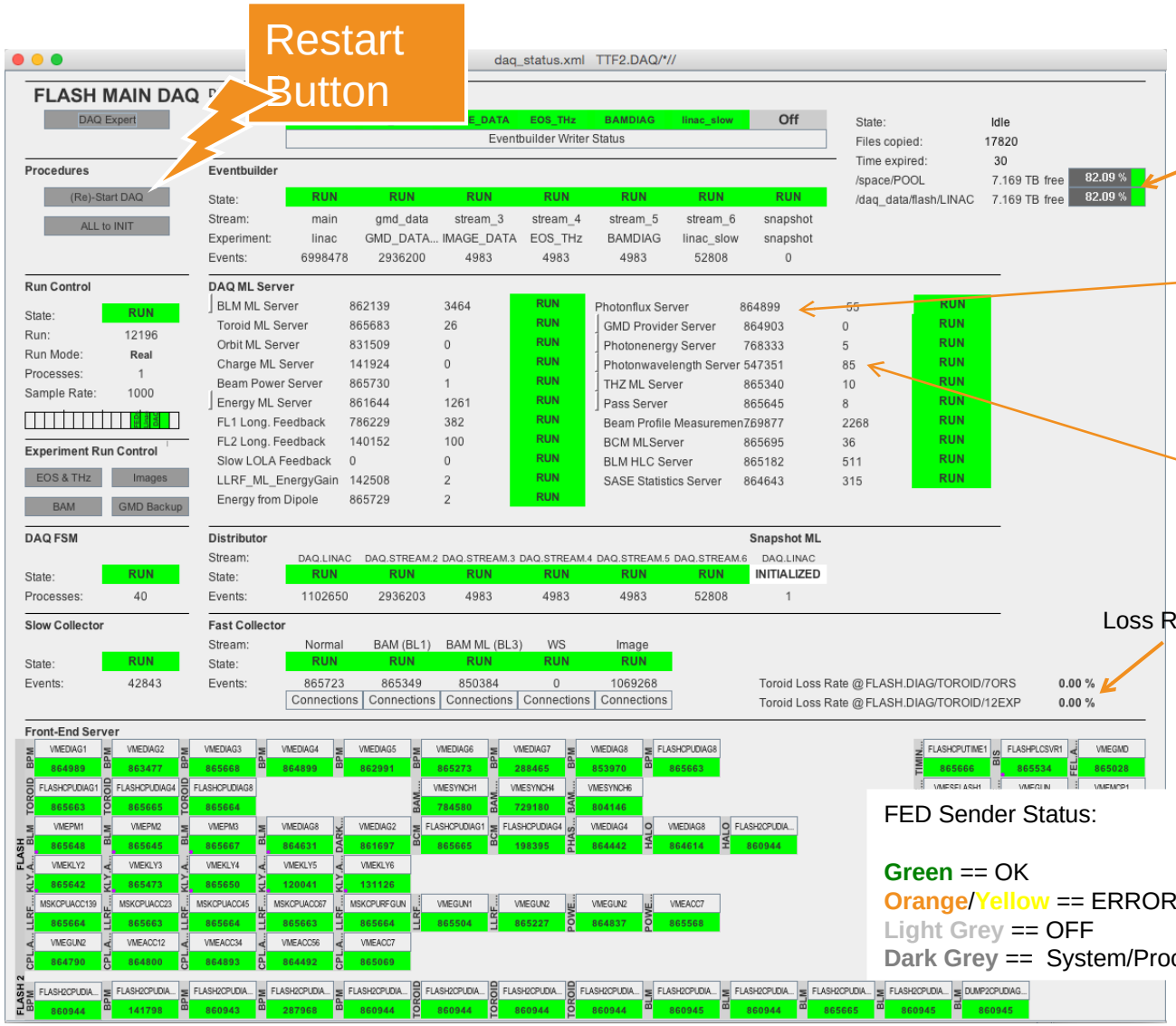
Event Builder and Writer

DAQ Middle Layer Servers

Buffer Manager

Collector

Front-End Server



Disk Status

Event Number should increment

Lost Event Number should **NOT** increment

Loss Rates

FLASH MainTaskbar -> Status -> Control System -> DAQ Status -> MAIN -> Status

Status

Accelerator Data Acquisition Instances

Status

- **Total number of DAQ instances at DESY(Zeuthen): 36**
  - XFEL (M), FLASH (M), FLASH Experiments (FS-FLASH-D), LUX/Angus (CFEL/UHH), REGAE (CFEL), laser systems (FS-LA), FLASH Forward (MPA/FTX), SFLASH project (MPY), AMTF and CMTB for cryogenic module testing (MHF-p/MSK), ALPS (FH), KALDERA (M) ...

XFEL: 10 million parameters, O(30k) channels in DAQ / **XFEL ,** O(4k) channels **FLASH**

- **Data rates and volumes (24/7 operations):**

	Input Rate [MB/s]	Disk Output Rate [	Write Rate to dCache [MB/s]
XFEL (all)	**~1200 - 1400**	up to 37 TB/day (~130 TB/d uncompressed)	200 – 400 400 400
FLASH (Linac only)	**160**	~ 3.5 TB/day	60

- **Data kept on disk** for XFEL (dCache /pnfs.desy.de/xfel-linac) for 2 weeks (about 800 TB space, raw 1 PB integrated into dCache cluster)
- **Data volume taped** so far (2022):
  - XFEL /pnfs/desy.de/xfel/real: 658 TB
  - FLASH /pnfs/desy.de/ttf2/real: 1.7 PB

Accelerator Data Acquisition Instances (cont'd)

Lessons Learned

Observations – Transport Protocol

- UDP Multicast / DOOCS DAQ push protocol efficient and scalable.
- Setup and configuration requires significant tuning and knowledge of Linux kernel.
- Watch out for Linux kernel developments and security patches – careful testing required.

Observations – Shared Memory and Buffer Manager

- Event structure tagged with pulse number is key ingredient for analysis.
- Excellent feature of having all shot-synchronized data in an event-like structure for online and offline processing.
- Monitoring made easy.
- Event history size is defined by hardware memory.
- Usable for "slow" software-based feedbacks and many high-level controls applications.
- Rapid application interface via MATLAB was not feasible.

Observations – Offline Analysis and Tools

- Missing metafile information for quick search – aka "fast browsing" desired.
- Good performance of proprietary file format, yet it prohibits using tools available at large in many communities – e.g. HDF5 conversion required first.
- Tools availability and functionality define community acceptance.

Observations – Instances

- Feature of parallel instances allows for redundant setup, testing and debugging.
- Flexible and quick ramp up of new instances.

Observations – Offline Storage

- Underestimated temporary space requirements.
- Insufficient cluster design.
- Upgrading intermediate storage space to 5 Pbyte system.
- Long-term storage uses dCache. Good experiences made here so far.
- Connection to HPC systems desired (GFPS) - possible but not standard.

Outlook

Revised Control System Concepts

XFEL, FLASH, ARD Projects and Test Facilities

- **Offline data management and analytics**
  - **Reduction**: no clever vetoing possible w/ accelerators but front-end reduction (FPGA level)
  - **Long-term storage** (taping or not) important for quality and performance evaluation
  - Simple and fast access of data is a must – browsing through Terabytes for interesting events -> tools work kind of but **retrieval concepts need to be revised**
  - Data does not have to be kept for paper publishing but for **documentation of machine performance and quality checks** (e.g. charge accumulation, cavity failures)
- **cw-operation** – High input rate with quasi-pulse like structure
  - **Dedicated Ethernet topology and big-pipe backbone** required for widely distributed systems – 10 Gbit/s switches at minimum at front ends
  - Online processing requires performant hardware – **revise shared memory design**
- **High availability of accelerator / operations (XFEL and FLASH)**
  - **Heavily extended monitoring** of any accelerator data - machine parameters as well as environmental (10 million @ XFEL) – current implementation needs to be extended
  - Shot-/train-synchronized **online data processing for failure detection and prediction**
  - Typical **use cases**: trends, post-mortem analysis, correlations, play-back of old data,…
  - Utilize data analytics and data science frameworks (AI) – **lack of resources**

References

References

Publications

- **Current and general overview of the accelerator control system as implemented at the European XFEL:**
  T. Wilksen et. al., “The control system for the linear accelerator at the European XFEL - Status and first experiences”, in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control System (ICALEPCS’17), Barcelona, Spain, Oct. 2017, pp. 1-5, paper MOAPL01
  http://icalepcs2017.vrws.de/papers/moapl01.pdf

  Paper on the accelerator data acquisition system as used at the European XFEL:
  T. Wilksen et. al., "A bunch-synchronized data acquisition system for the European XFEL accelerator”, in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control System (ICALEPCS’17), Barcelona, Spain, Oct. 2017, pp. 958-961, paper TUPHA210
  http://accelconf.web.cern.ch/AccelConf/icalepcs2017/papers/tupha210.pdf

  Paper on high-level controls as implemented at the European XFEL:
  L. Fröhlich et. al., “High level controls for the European XFEL”, in Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control System (ICALEPCS’15), Melbourne, Australia, Oct. 2015, paper MOPGF101
  https://accelconf.web.cern.ch/ICALEPCS2015/papers/mopgf101.pdf
- **Paper on virtual XFEL based on the DOOCS DAQ:**
  W. Decking et al., “The Virtual European XFEL Accelerator”, in Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control System (ICALEPCS’15), Melbourne, Australia, Oct. 2015, paper TUD3O04.
  https://accelconf.web.cern.ch/AccelConf/ICALEPCS2015/papers/tud3o04.pdf
- **Short overview on tools for DAQ data retrieval (slightly out-of-date):**
  V.Rybnikov et al.,“FLASH DAQ data management and access tools”, PCaPAC’10, Saskatoon, Canada (2010), pp. 195–197.
  https://accelconf.web.cern.ch/pcapac2010/papers/frcoma02.pdf
- **Paper on the implementation of the inner workings of the DOOCS DAQ (slightly out-of-date):**
  V. Rybnikov et al., “A Buffer Manager Implementation for the FLASH Data Acquisition System”, PCaPAC 2008, Ljubljana, Slovenia, October 2008
  https://accelconf.web.cern.ch/pc08/papers/tup010.pdf
- **Original paper of the overall DAQ concept (slightly out-of-date):**
  A. Aghababyan et al., “Multi-Processor Based Fast Data Acquisition for a Free Electron Laser and Experiments”, in IEEE Transactions on Nuclear Science, vol. 55, No. 1, pp. 256-260, February 2008.
  https://ieeexplore.ieee.org/document/4382853

Thank you

Contact

DESY. Deutsches Elektronen-Synchrotron

www.desy.de

Tim Wilksen
MCS
Tim.Wilksen@desy.de
3456

Vladimir Rybnikov
FTX
Vladimir.Rybnikov@desy.de
4846