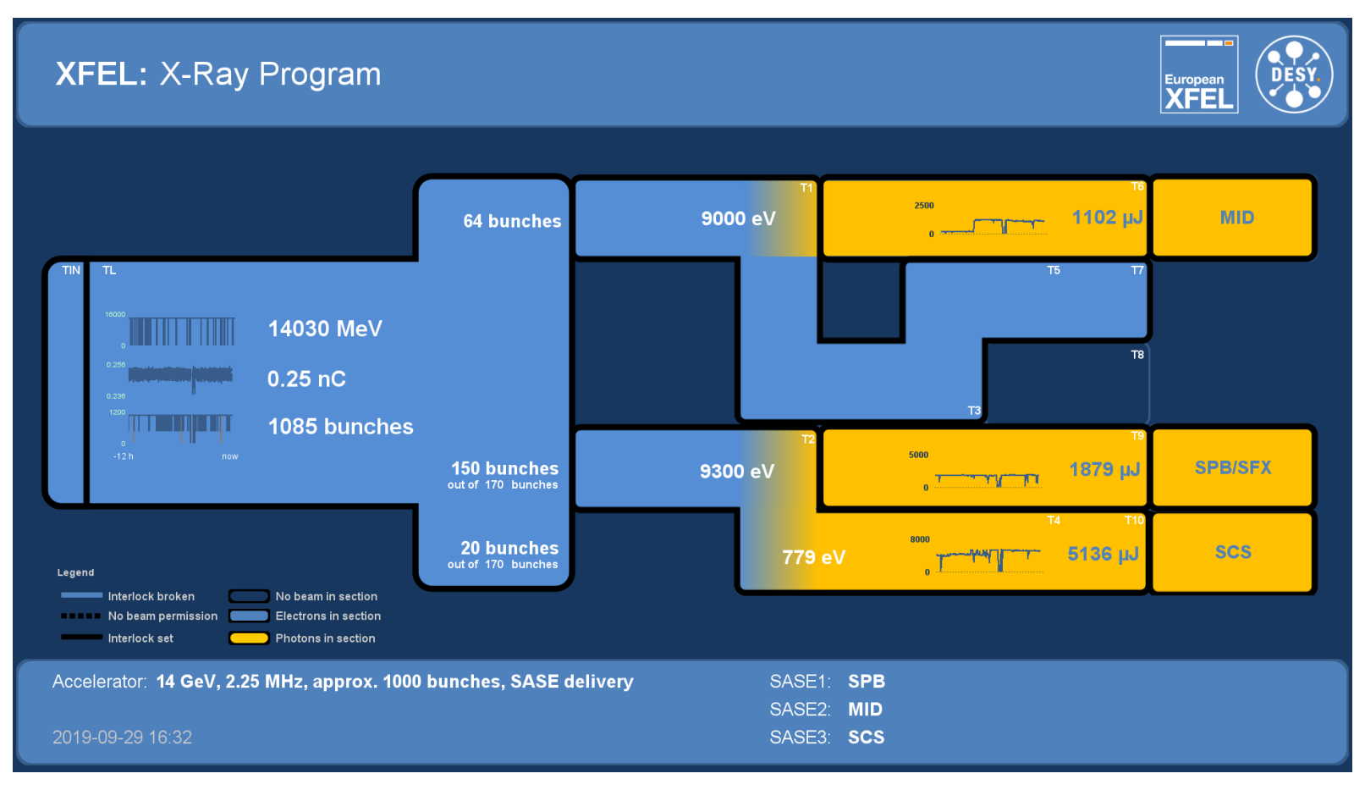
Revisiting the Bunch-Synchronized Data Acquisition System for the European XFEL Accelerator

Tim Wilksen*, Arthur Aghababyan, Lars Fröhlich, Olaf Hensler, Raimund Kammering, Kay Rehlich, Vladimir Rybnikov (DESY, Hamburg)

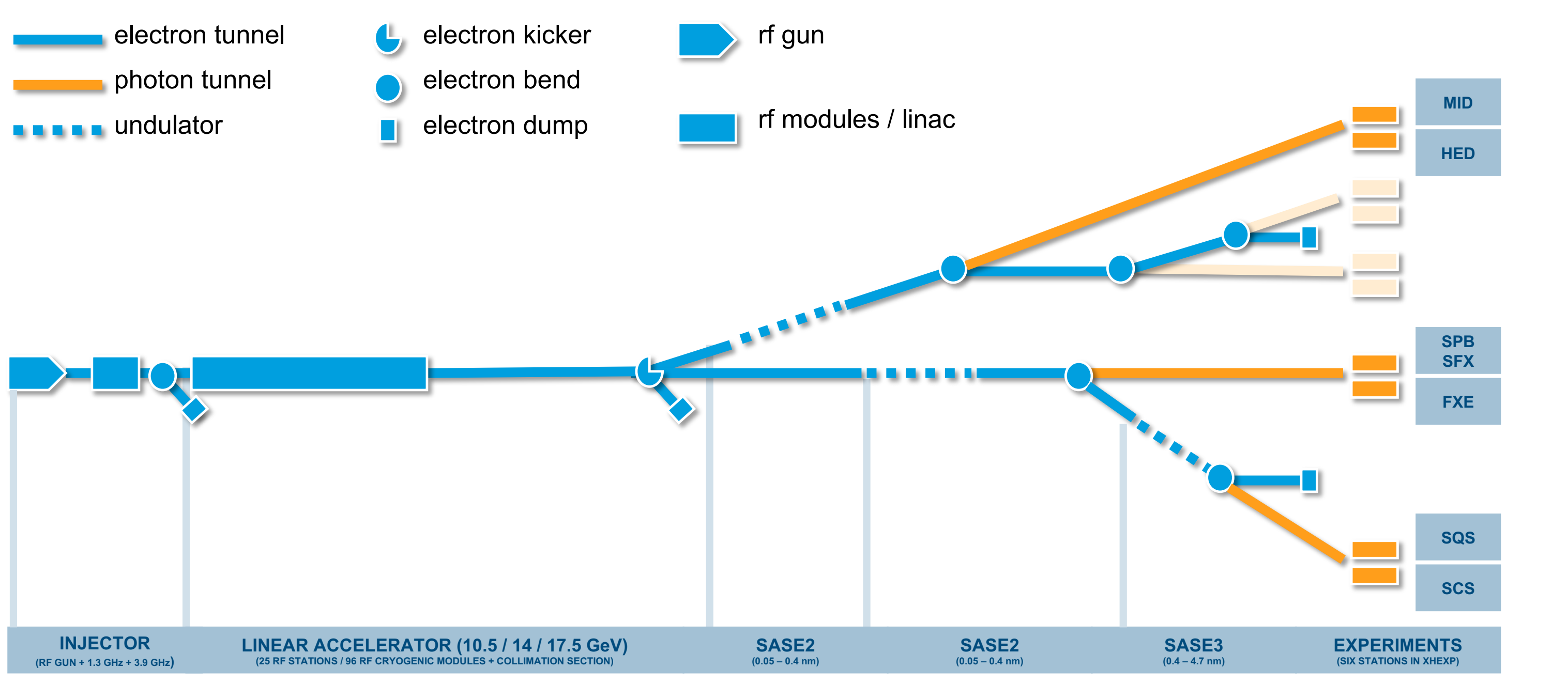
Abstract

After about two years in operation the bunch-synchronized data acquisition as used with the accelerator control system at the European XFEL is being revisited. As we have now gained quite some experience with the current system design it was found to have some shortfalls - specifically as for offered methods and tools for data retrieval and management. In this paper the current implementation is being discussed and taken as an evaluation of new frameworks readily used by many internet and business companies in the context of modern data collection and management technologies. The main focus is currently put on streaming technologies which are being reviewed with respect to feasibility and adaptability for control system architectures at DESY's accelerator facilities.



Public European XFEL Accelerator Status Panel

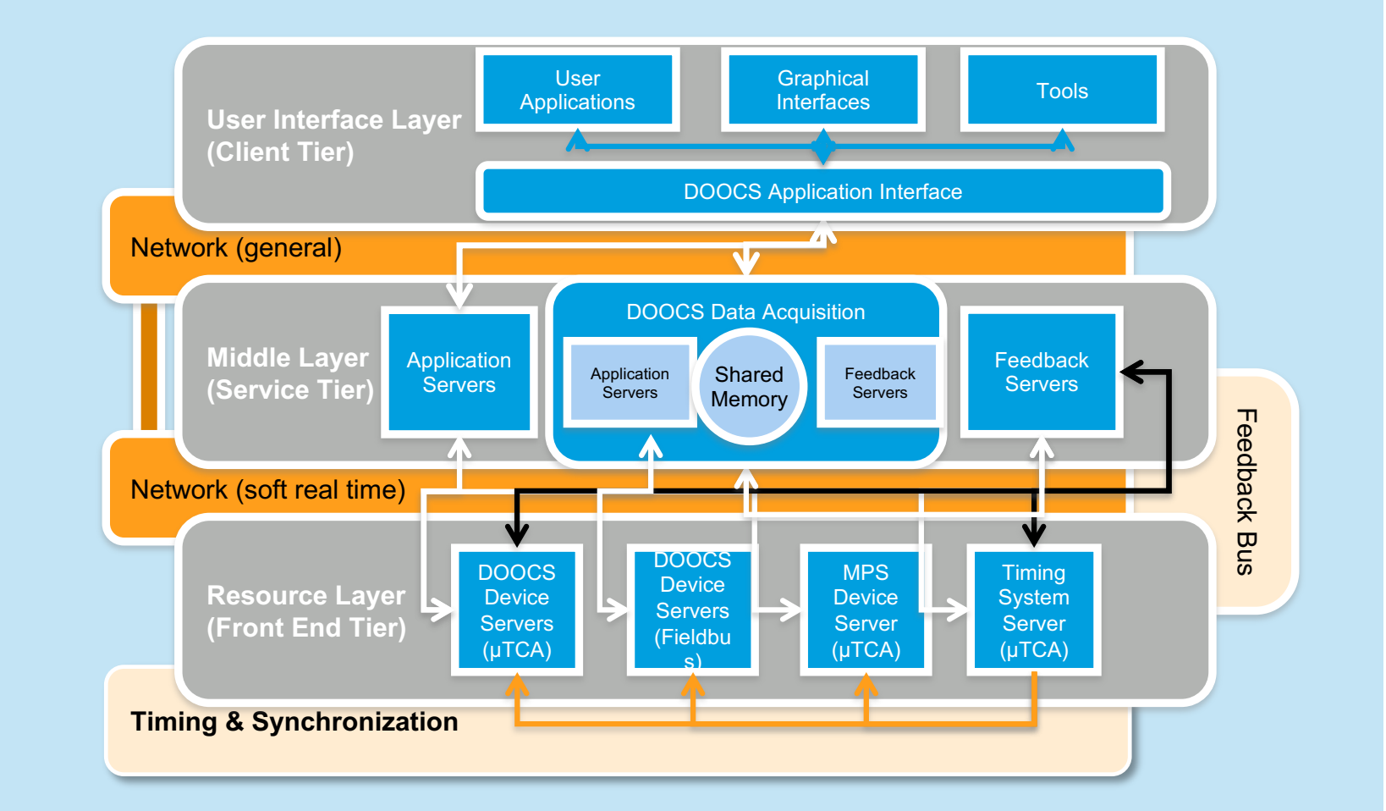
The European XFEL Accelerator – Beam Line Layout



Specifications
Superconducting linear accelerator with beam energies 10.5 GeV / 14 GeV / **17.5 GeV**
Pulse repetition rate at **10 Hz** with **27000 bunches/s**
3 Photon beam lines with 6 experiment stations
Photon energy 0.3 – 24 keV
Photon pulse length 10 – 100 fs
Photon Pulse energy ~ mJ

Parameter	
Electron beam energy	10.5 GeV / 14 GeV / 16.5 GeV
Bunch charge	0.02 – 1 nC
Peak current	2 – 5 kA
Slice emittance	0.4 – 1.0 mm mrad
Slice energy spread	4 – 2 MeV
Shortest SASE wavelength	0.04 nm
Pulse repetition rate	10 Hz
Bunches per pulse	2700
RF-Pulse length (flat top)	600 μs

Accelerator Control System

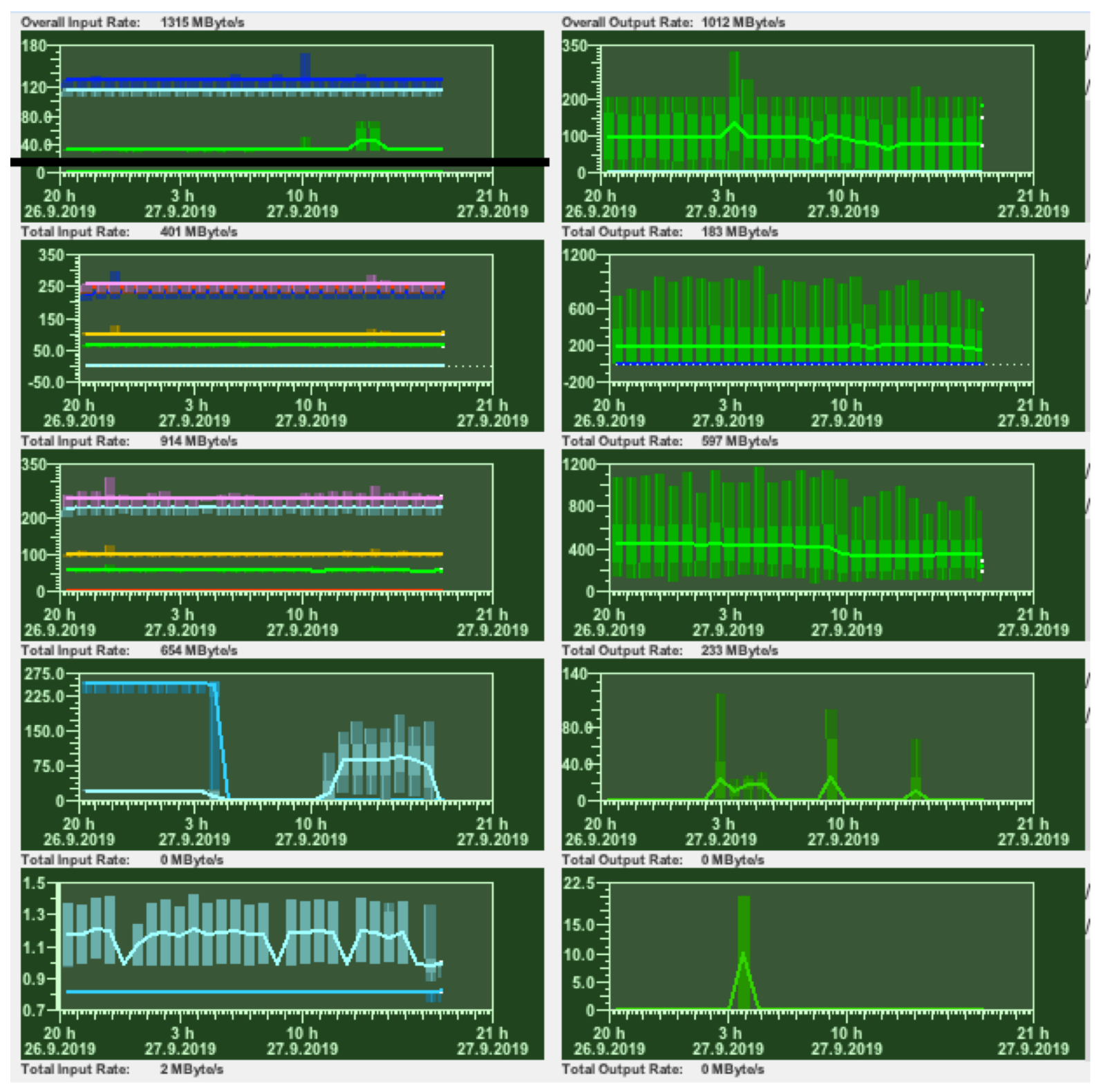


Accelerator Control System layout with integrated DAQ

On the device layer MicroTCA-based ADC modules, camera devices as well as PLC and other embedded devices are sending data to collector processes. A fast collector acquires data at the bunch repetition rate from triggered devices and a slow collector polling the data at rates of about 1 Hz from other hardware. The data has been tagged on the front-ends with a unique shot number provided by the timing system. Both collector processes are feeding the received data into a buffer manager using shared memory for storing the data. The distributor process functions as buffer manager and is in charge of managing the shared memory structure. Middle layer processes can connect to the buffer manager and read and/or write back to it. Once all data for a given shot number has been acquired in shared memory it is sent to the event builder and –writer processes. They will write the data as compressed files to disk. To tape it the dCache [5] facility at DESY is being utilized. Several applications interfaces exist to read the data files and extract data for individual control system parameters (i.e. DOOCS channels on the front-end server). Middle layer server can provide data for other applications including graphical user interfaces for online monitoring purposes.

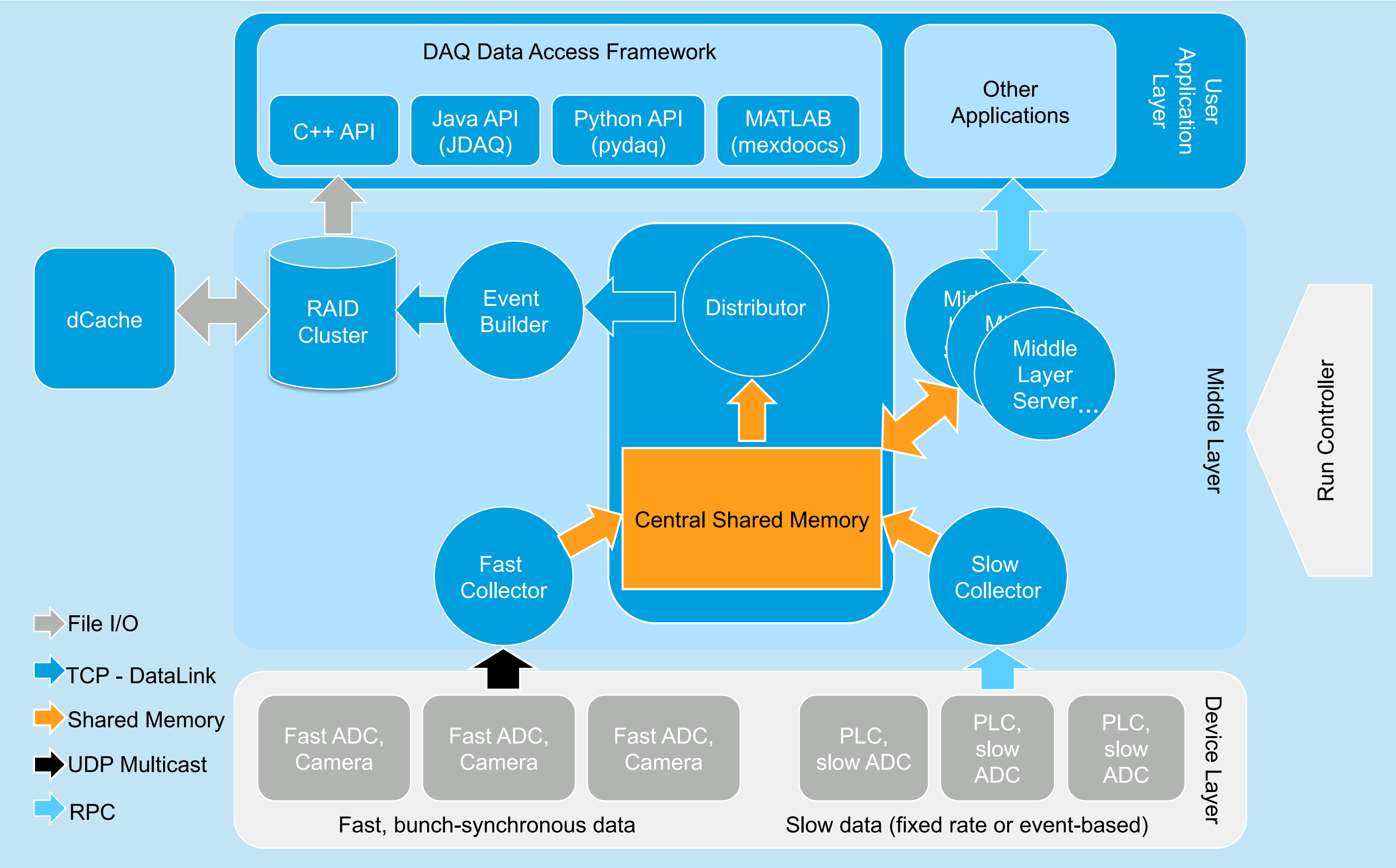
Status and Operation

Parameter	
Total input rate (24/7)	2.1 Gbyte/s
Compressed storage rate	Up to 30 Tbyte/day
Total number of channels	~ 100 k
Number of DAQ instances	6 + 2
Number of streams per instance	2 - 8
Temporary storage space	Up to 1 Pbyte (dCache)



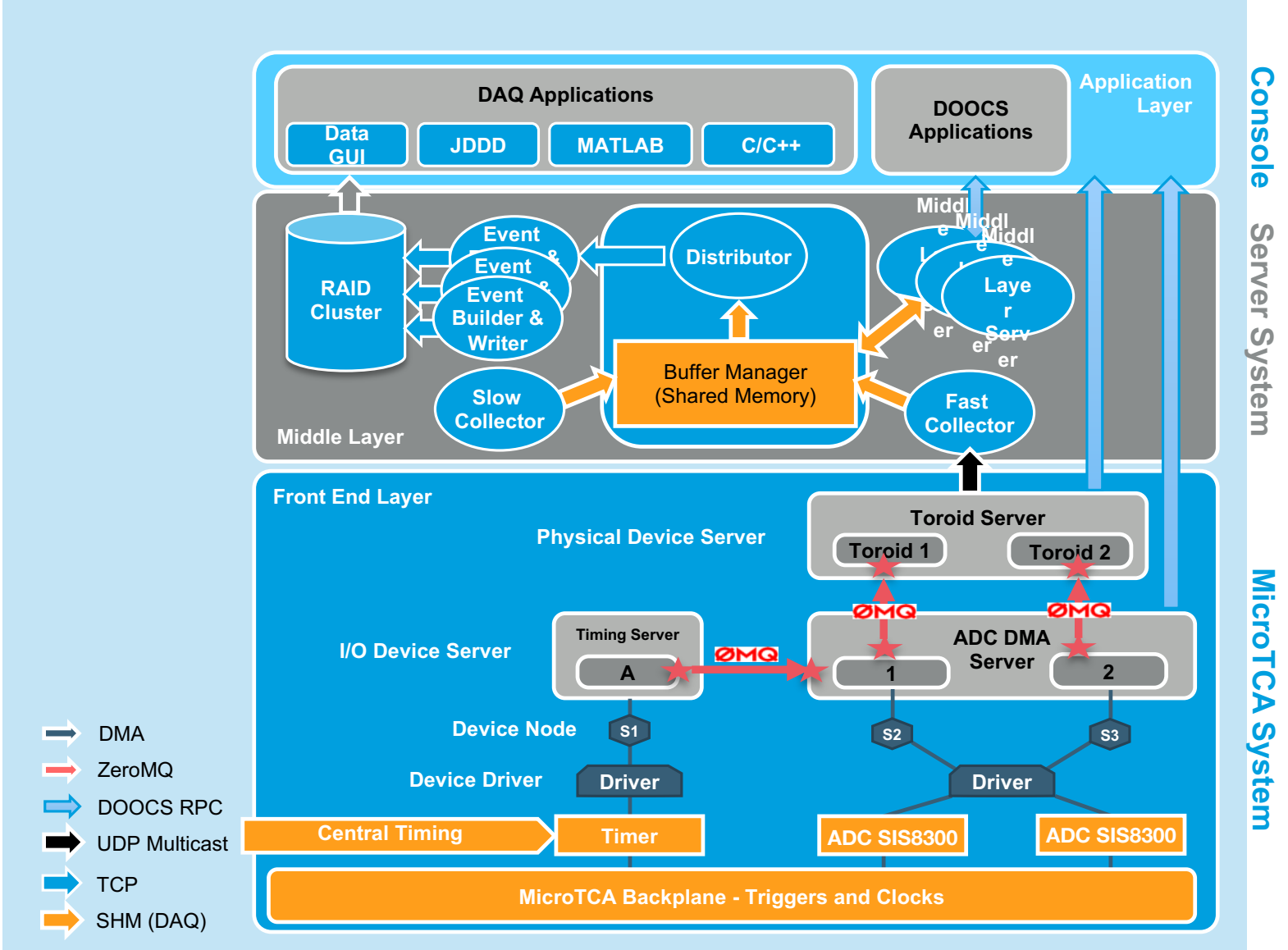
Input rates (left) and output rates (right) of DAQ instances

Schematic Overview of the Accelerator DAQ



Data acquisition instance integrates into middle layer server of overall accelerator layout

Data Flow from MicroTCA Front-End to Applications



Timing System and its pulse or shot number

- 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.

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 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 1 Pbyte system.
- Long-term storage uses dCache. Good experiences made here so far.
- Connection to HPC systems desired (GFPS) - possible but not standard.

Outlook

EuXFEL cw-Operations

PETRA IV Project

Features for Next-Gen DAQ

- Event structure with shot-synchronized and bunch-resolved data requiring corresponding FE support (timing and synchronization).
- Streaming concepts for efficient data collection from FE and other sources which allow online processing.
- Near-real time access of shot-synchronized data at stream level.
- Control system framework independent, i.e. use plug-in mechanism for FE software (sender) and given control system API.
- Metafile information database or catalog to provide tagging information and allow for categorizing system events like cavity trips, certain kinds of beam loss, SASE intensity loss or any kind of post-mortem events defined by a trigger rule.
- Browser tools to access data with tagging functionality for event types (e.g. cavity trip, beam loss) – “quick search”.

* Tim.Wilksen@desy.de