ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA161

REVISITING THE BUNCH-SYNCHRONIZED DATA ACQUISITION SYSTEM FOR THE EUROPEAN XFEL ACCELERATOR

T. Wilksen†, A. Aghababyan, L. Froehlich, O. Hensler, R. Kammering, K. Rehlich, V. Rybnikov
Deutsches Elektronen-Synchrotron (DESY), Hamburg, Germany

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 the offered methods and tools for data retrieval and management. In this paper issues of the current implementation are being discussed and taken as an input for 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.

INTRODUCTION

The European XFEL is a 3.4 km long X-ray Free-Electron Laser starting in Hamburg, Germany, and ending south of Schenefeld, a city in the neighbourhood of Hamburg. The linear, super-conducting accelerator comprises a photocathode laser-based RF gun followed by a 1.6 km accelerating linac section with 96 superconducting cavity modules installed. The electron energy operating points can be chosen between 11 GeV up to 16.5 GeV with present number of RF stations. The machine can provide up to 2700 electron bunches for each shot at a repetition rate ranging from 1 Hz to 10 Hz. The bunch repetition rate can vary between 100 kHz and 4.5 MHz. After a collimation section in the main tunnel the electron bunches are distributed according to a highly configurable bunch pattern selection into two electron beam lines with each one resp. two undulator sections to produce X-ray photon beams for six possible experiment stations as shown in Fig. 1. Currently the full machine is operated with up to 1000 bunches in the main linac and up to 400 bunches each in the two electron beam lines.

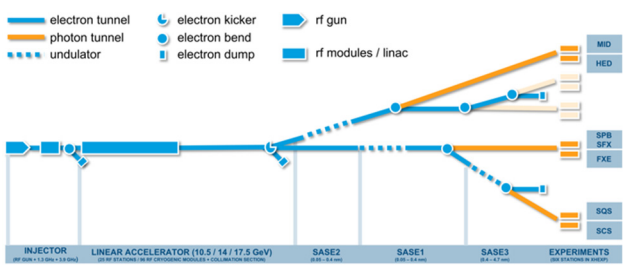

Figure 1: Layout of the European XFEL accelerator.

ACCELERATOR CONTROL SYSTEM

General Layout

The accelerator control system for the European XFEL had been designed using a standard hardware platform based on the MicroTCA.4 technology [1] and a software framework suitable to control a pulsed linear accelerator driving a free-electron laser. As laid out in [2] the choice made uses primarily DOOCS [3] as well as TINE [4] which is integrated by default into the DOOCS core libraries. One of the key requirements was to be able to synchronize all accelerator beam diagnostics data and RF- resp. LLRF information for diagnostic displays and high-level controls or physics applications. For this the accelerator control system layout integrates shot-synchronized and bunch-resolved data acquisition instances as part of the overall control system. This is illustrated in Fig. 2.

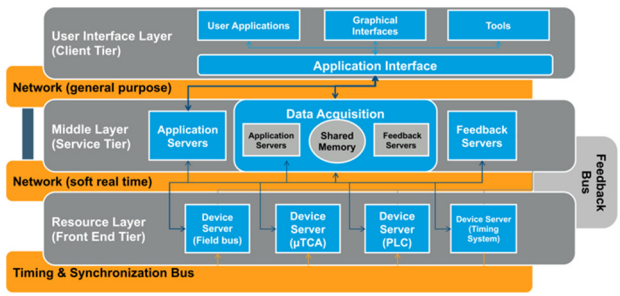

Figure 2: Accelerator control system layout with three layers and integrated data acquisition system on the middle layer.

The integral nature of the data acquisition part within the control system allows for synchronized and efficient online access of all shot-related data. This provides the possibility to plug in services on the middle layer using the shot-synchronized data for high-level applications, doing online processing and computations or even providing slow feedback capabilities. Those services can work seamlessly together with classic middle-layer applications using standard DOOCS calls to retrieve the data.

On the user interface layer the shot-synchronized data is available through dedicated interfaces alongside with data from any other DOOCS channel. Though there is a limitation on the amount of data and its selection with respect to the time range, it is possible to visualize these together with archived DOOCS data.

†Tim.Wilksen@desy.de

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA161

DATA ACQUISITION

Design and Layout

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. A schematic overview is shown in Fig. 3.

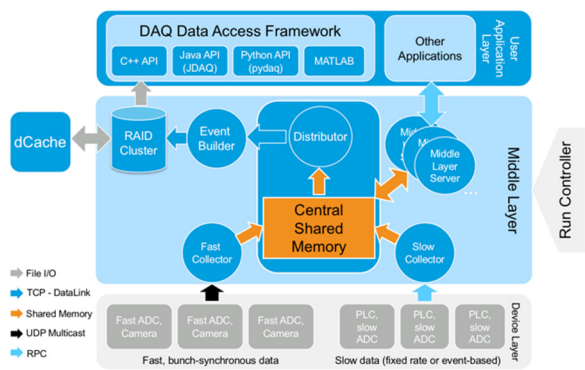

Figure 3: Schematic layout of a single data acquisition instance as used for acquiring shot-synchronized data from MicroTCA systems at the European XFEL linear accelerator control system.

The following sections will explain the individual components of this design.

Front-End Devices

Providing a unique identification of individual pulses or shots is achieved by sending timing system information i.e. clock signals, trigger events etc. including a unique number created by the timing system master to all front-end devices. For the European XFEL accelerator control system this has been implemented using a timing system based on the MTCA.4 standard. Since all front-ends for beam diagnostics, RF- and LLRF-controls are using the MicroTCA hardware platform a MTCA.4 module has been developed at DESY together with the University of Stockholm. This module is installed in every MicroTCA system. It receives the timing system information from the master via a fiber optic link at the shot repetition rate of 10 Hz. Clocks, event trigger and shot number are then provided to the MicroTCA system components via backplane as a PCIe interrupt or via M-LVDS lines. A timer server program on every MicroTCA system provides the timing system information also via ZeroMQ [6] to other applications as shown in Fig. 2. The stamped data is sent via a push-type protocol based on Multicast UDP to all subscribers. That way multiple clients can connect to front-end devices without creating an additional load to the front-end systems. The instance on the front-end to subscribe to is called a “sender”.

Collector

A collector process is running on a dedicated server node, usually a multi-core server with sufficient memory[1]. It subscribes to all front-end senders via Multicast, which are capable of acquiring data at bunch repetition rates. Hardware offering control system parameters of the accelerator without being connected to the XFEL timing system are retrieved via standard DOOCS RPC calls and stamped with the current shot number.

The data from an individual Multicast UDP sender is packed up into a so called “sender block” which combines all its DOOCS channels to be sent to the data acquisition system with a server block header. The header contains name, length, timestamp, number of channels, status and shot number to identify it. The collector allows for several retries until it would mark this specific server block as missing for this shot.

The collector process is registered as a client to the buffer manager and permitted to write to the shared memory area to store the server blocks.

The number of collector processes is not restricted. Several collector instances can connect to the same buffer manager. One individual collector process may have different locations for separating the received channels into smaller subsets.

Shared Memory

The buffer manager [7] is the master of managing the shared memory area. It allocates the overall memory structure separated into a “client segment” for keeping all the client information subscribed to the buffer manager and an “event segment” for tracking the information about current shot data available in the shared memory. The data section itself holds all server blocks in memory for a couple of shots. These control and data segments are being set up according to a run control configuration. This configuration is aware of all sender locations of which data is to be expected and acquired.

Clients can subscribe for reading to all server blocks managed by the buffer manager or by specifying just a subset of what they are interested in. Clients can also write to the buffer manager like the collector processes do when filling the shared memory with received server blocks. Whenever expected server blocks for a given shot number

[1] Currently up to 512 GB RAM is used for DAQ nodes.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA161

are stored interested clients are being notified that the data is now available. DOOCS middle layer server with an interface to the buffer manager can subscribe to specific server blocks available in shared memory, too. Examples for this case are server processing shot-synchronous data of several front-end sender locations for displays e.g. the electron beam orbit derived from all BPM devices; or the energy measurement of the overall beam energy using LLRF data from all the RF stations. These middle layer servers can even write back the results of their computations to shared memory, e.g. the energy, and add it to the current shot data still in memory. The shot data is considered to be complete only if every one of the subscribed clients has been notified about the availability of the data. Every client must signal having received the data and all potential producers must have written back their data. Subsequently, the event record will be released from the shared memory after a waiting time. There are several mechanisms in place in case one of the clients is not present anymore, server blocks are missing or sender and clients are just very late in providing data. It is guaranteed that for every shot number an event record is created even if individual sender blocks are missing i.e. are potentially empty.

The distributor which hosts the buffer manager is furthermore able to group selected server blocks together to various event records and distribute them into data sets or so-called "streams". The configuration which server blocks form an event record and correspondingly a stream is defined by the run control configuration. This stream configuration is quite useful for separating groups of channels logically into manageable data volumes. It allows also to dynamically switch on or off sending individual stream data to be further processed and archived on disk.

The event builder process connects to the distributor and receives a stream data via a dedicated TCP data link. This stream data is then distributed further to one event writer process per stream, which writes eventually the data files to disk. From there the data can be written to dCache disks and eventually – on request only – being taped for long-term storage.

Applications and Tools

To access the data file on disk several interfaces exist. There are API libraries available for C/C++, Java, MATLAB and Python. Several tools written in C/C++ and Java exist to just browse available data on disk and display it quickly e.g. the DAQ data GUI. Tools for extracting only a subset of the data help reducing the amount of data collected in a given file.

In general, there are two ways of accessing these offline data files. Either via direct disk or filesystem access using the DAQReader library for which the file system needs to be mounted or via a farm of data servers. The latter approach is done by sending an XML-type request containing the time range for the desired data, the stream and the names of the DOOCS channels to the DAQ data server farm. The requested data is extracted from the files and send back to the client. This way one the DAQ data server takes care of finding the correct file sets and extracting the desired channel data.

The standard graphical user interface used for the DOOCS control system framework, JDDD, is able to read DAQ data and visualize it together with data from local DOOCS archives. For this the plot widget supports the selection of a time range in an archived data plot for which one would like to retrieve the data again through the DAQ data server. Essentially, the same XML request is sent to the server farm as with the other interfaces. The received data is displayed together with the archived data.

STATUS

In the following paragraphs the current status of the DAQ system as well as some lessons learned will be described.

Operation

As of now the European XFEL accelerator control system comprises six DAQ instances[2], one more is being set up currently and one instance is for test purposes. The first and main instance is used for collecting all electron- and some photon-beam related diagnostics data, while the next two are collecting LLRF cavity and RF coupler interlock information. All three instances are used for online monitoring of orbit, charge transmission, beam power and slow RF feedbacks as well as for (beam) energy management and are therefore critical for accelerator operations. These DAQ instances are running 24/7. Another DAQ instance is currently used for collecting data from the photocathode laser system, mainly for performance and diagnostic purposes. For the optical synchronization subsystem one more instance is in the process of being setup currently. Here it is worth to notice that some of the data is actually not syncrhonized with the 10 Hz shot rate but is recorded continuously independent of the shot trigger. For the virtual XFEL – a testbed for testing and evaluating mostly high-level controls software relevant for accelerator operations but also for software checks in general – has been set up separately from the accelerator controls. And finally, another test system can be configured like one of the above systems to test new configurations, software or new algorithms.

While the overall accelerator control system provides about 10 million control system parameters including meta parameters like polynomial parameter settings for archive filters or channel descriptions, only a small subset is fed into the DAQ instances. It should be noted, that channels as appearing in the DAQ systems are usually hybrid channels which can combine several DOOCS parameters into one. The combined input rate of these instances is about 2.1 GByte/s, not yet taking into account the data from the optical synchronization and test instance.

As for data being written to disk for a later analysis we do collect about 30 TByte/day as compressed data[3]. This data is moved to the dCache disks, usually kept there for up to two weeks and then removed unless it has not been requested to put it into the long-term storage.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA161

All these instances usually are up and running 24/7 and are only interrupted on maintenance days or in longer shutdowns. An overview panel gives a quick glance of the overall data transfer status as illustrated in Fig. 4.

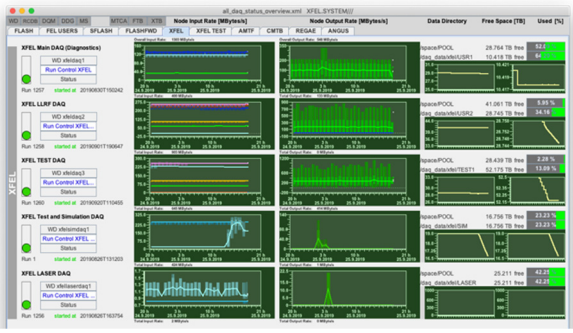

Figure 4: Overview of input (middle left column) and output rates (middle right column). On the rightmost side the disk size for transfer space is being monitored.

Using the stream configuration mechanism, a dedicated one for the electron beam diagnostics data is being provided as well as one for parts of the photon beam diagnostics data. Information about the RF cavities and LLRF controls is combined together with RF coupler interlock data. Here the streams have been split up into two halves to cope with the large number of files being written and to manage file handling with respect to dCache. It is possible to switch on or off individual distributor locations (input streams) to control even further which RF station data is actually fed into the streams. Image data is fed into separate streams to simplify the analysis.

Observations - Transport

While UDP multicast enables the accelerator control system to collect in a very efficient way the information off the front-end systems, its configuration at various levels turns out to be tricky. This starts with setting up the Linux kernel correctly to cope with high amounts of UDP traffic, configuring Ethernet adapters and drivers so that one can use the full available bandwidth and use a proper segmentation with respect to the network topology. Initially set up interfaces with link aggregation seemed to be not as performant and reliable as expected, missing a proper LACP set up of the system. Therefore, adapters were split up according to the subnet topology which fits the collector architecture much better but is more problematic for network routing and its topology. The current configuration however works very reliable and scales nicely.

One major issue arose early this year using Linux kernel versions with patches to mitigate Spectre and Meltdown vulnerabilities showing up late last year. A dramatic performance loss was noticed when collecting the data via UDP using those kernel versions. Still early 5.x versions had massive impacts while a version from fall last year without the newer Spectre patches was performing well. This was at first not easy to debug because it did not show up in our initial tests without beam during shutdown.

Observations - Instances

The UDP multicast-based data collection - it is implemented as a push-protocol - allows to run multiple DAQ instances in parallel subscribing to the same groups of multicast sender. This is very helpful when setting up, re-configuring, testing or debugging an individual instance because the original can be kept running while working on its twin system without disturbing operations. It allows further for better segmentation of offline data and offers more flexibility when managing files for individual streams. And last but not least, it can provide redundancy.

Observations – Shared Memory

The shared memory used by the buffer manager is *the* source of shot-synchronized and bunch-resolved data. This is useful for online monitoring of accelerator operations. At the European XFEL this feature is particularly used to display and track the orbit of the electron beam as well as the quality of the beam transmission derived from the transported charge and the beam loss monitor system. Utilizing the data from the RF station controls an energy manager allows for adaptive management of individual stations to provide the desired electron beam energy.

A couple of so called “slow-feedback” managers have been implemented on the main instance using the shared memory data. These can act on deviations observed either by the orbit monitoring or by looking at the bunch compression monitoring and acts on the RF setup as well as orbit steering. It is segmented according to the machine layout, therefore several of these feedback managers are running in parallel.

One drawback of these high-level control applications is, that they have to be written in C++ and have to use the API to access the shared memory and to register with the buffer manager. This kind of prohibits rapid-application development of accelerator physics tools as often desired in the accelerator control room by machine physicists. A similar request often appears at beamline experiments, where experimenters want to look online at data to be recorded while still changing the set-up of the experiment. This is often referred to as near-real-time online monitoring. Since the accelerator data acquisition presented here in detail has been in use at FLASH beamline experiments since quite some years now, this request has come up here, too. One solution for now is to use a standard DOOCS server which can access the shared memory and provide data from any channel in there via subscription. While this works quite well, not many experiences have been made to which extent this will scale. Likely, this approach cannot cope with large numbers of subscribed channels.

Observations – Offline Storage

Many issues have been observed with storing data on local disks of the DAQ nodes itself but also with shipping it to the DESY dCache instance. Most of them were newbie mistakes, some have been conceptual issues. It turned out to be problematic to use DOOCS archives together with DAQ data on the same system. This affected moving the

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA161

data to the dCache disks. In general, our DAQ node hardware can only keep the offline data for a few days therefore the dCache disk space is the offline storage area. To resolve restrictions with the current setup a new 1 PB dCache space will be deployed in the near future. Connected dynamically to several dCache nodes, this will hopefully speed up our shipping process.

Observations – Offline Analysis and Tools

One of the longstanding complaints had been the somewhat tedious use of the offline data files. The choice to implement a proprietary binary format for the data files had been made early on in the development of the DAQ system, since it was found to be the most efficient one when looking at the compression rate and write speed. Investigations of the ROOT file format and later HDF5 did not meet our expectations with respect to its performance at the time. However, the proprietary format reduces significantly the number of available tools to retrieve and analyze the data. One remedy is to convert the data into the HDF5 format later in batch. The real issue here might be a perceptional one, in so far, that 30 TB of data per day is not as easy to "browse" as many control system users would expect from using archiver and small databases. Clearly missing is here some kind of metafile database which holds all the meta data to find quickly the information one is looking for.

One should point out here, that the event-like structure of the data and the tagging of all data with a unique macro pulse number turned out to be a key element to ease the offline analysis. There is no hassle at all anymore to match timestamps and such. With a metafile system to search easily for individual events according to given categories, this would complement it.

Next-generation DAQ

Considering the aforementioned observations and shortcomings, the following requirements are considered to be key to the next-generation accelerator DAQ. A new DAQ type is foreseen to serve the European XFEL as well as future accelerator projects like PETRA IV.

a) Event structure with shot-synchronized and bunch-resolved data requiring corresponding FE support (that is, timing and synchronization information has to be present on the FE).
b) Using streaming concepts for efficient data collection from FE as well as any other sources and processing capabilities within the streaming architecture for online analysis.
c) Near-real-time access of shot-synchronized data.
d) Control system framework independent i.e. use plug-in mechanism for FE software and the used control system framework.
e) Metafile information database to allow for tagging and categorizing events. E.g. cavity trips, certain kinds of beam loss, SASE intensity loss or any kind of post-mortem events defined by a trigger rule.
f) Browser tools to access data with tagging functionality for event types (e.g. cavity trip, beam loss).

First attempts to use Apache Spark and Kafka frameworks for evaluation have been started already taking part of the above requirements into account.

CONCLUSION

The DOOCS-based shot-synchronous and bunch-resolved data acquisition system has turned out to be an essential part of the overall accelerator control system for operations and diagnostics of individual subsystems. While the online monitoring features are performing quite well, the capabilities of analysing the offline data are not yet explored and utilized to the level which would be desired. I.e. to allow for in-depth analysis of individual subsystems as well as the accelerator machine as a whole. Key issues like availability and reliability are crucial now for the European XFEL as it has turned into a user machine. To enable a better understanding of the machine to improve the performance e.g. by optimization the SASE process as well as reducing and preventing faults in soft- and hardware, collecting all relevant data and processing it for analysis is essential.

A major revamp of the data acquisition system is foreseen based on the experience made with the current implementation but will also consider utilizing new technologies nowadays readily available within the data management and analytics field.

REFERENCES

[1] H. Schlarb *et al.,* "The case of MTCA.4: Managing the introduction of a new crate standard at large scale facilities and beyond," in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control System (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper FMOPPC081, pp. 285-287.

[2] Distributed Object-Oriented Control System, DOOCS, http://doocs.desy.de/.

[3] 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. doi:10.18429/JACoW-ICALEPCS2017-MOAPL01

[4] Three-fold Integrated Network Environment, TINE, http://tine.desy.de

[5] dCache, https://www.dcache.org

[6] ZeroMQ, http://zeromq.org/.

[7] V. Rybnikov *et al.*, "Buffer Manager Implementation for the FLASH Data Acquisition System", in *Proc. 7th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'08)*, Ljbuljana, Slovenia, Oct. 2008, paper TUP010, pp. 102-104.