EVOLUTION OF THE FLASH DAQ SYSTEM

A. Agababyan, G. Grygiel, O. Hensler, R. Kammering, V. Kocharyan, L. Petrosyan, K. Rehlich, V. Rybnikov, T. Wilksen[#], DESY, Hamburg, Germany

Abstract

The Data Acquisition System at the Free-Electron-Laser Hamburg (FLASH) has evolved since its implementation in 2005 into a reliable and versatile system, used for accelerator operations and studies along with a multitude of different photon experiment users, recording about 14 TB in 2008 for experiments only. Recently the DAQ system has been successfully upgraded with new hardware to accommodate increasing demands of beam line experiments, upcoming R&D work at FLASH i.e. for the ILC, and to prepare for the upgrade of the FLASH facility this year.

This paper describes the evolution of and experiences with the FLASH DAQ and highlights the key elements of its design to facilitate an expandable yet easily to duplicate system implementation.

MOTIVATION

The Free-Electron-Laser at Hamburg (FLASH) is the successor to the TESLA test facility (TTF) and the TTF VUV-FEL, which had been operated until 2002. FLASH has been designed to be used not only as a source for pulsed laser light in the extreme ultraviolet and soft X-ray regime, but also as a test bed for exploring new superconducting accelerator technologies for the European XFEL and the International Linear Collider.

Therefore a FLASH control system has to serve two communities - accelerator physicists and photon scientists - using the facility in quite different ways. It has to ensure stable and reliable production running for users on the one hand, and on the other, provide enough flexibility for frequent modifications of the beam line stations and their experimental setup. Accelerator R&D work needs usually exclusive use of the facility for dedicated study time. Properly recording and documenting the machine status as well as configurations throughout the R&D time should accompany this. To support all these demands, a data acquisition system (DAQ) has been integrated into the DOOCS-based control system [1], [2].

The DAQ system has to handle a large amount of data coming from the accelerator control and diagnostics devices as well as from the user experiments. It has to acquire and store all relevant information needed for accelerator operations, R&D work and photon beam experiments. More important has been the integration of accelerator and user experiment information to enable data analysis across machine and experiment. This can only be done, by synchronizing the various data sources. A timestamp might not be sufficient when looking at bunch-by-bunch information ore even at individual bunches. Tools and applications have to be provided for retrieving and processing the recorded data.

[#]Tim.Wilksen@desy.de

BASIC CONCEPTS

The above mentioned requirements and specifications resulted in the following key concepts while designing the DAQ system.

Bunch-Synchronized Data

Data from the VME-based ADC, with sampling rates at 1 MHz up to 2 GHz, is sent via multicast protocol to a fast collector. This is called the *fast channel* data because the ADC data sampling and collection is triggered with every macro pulse or shot. This kind of data is usually a spectra type with up to 2048 measurements, respectively for the newer ! TCA platform with up to 64k data points.

Slow channel data can be either collected from slow ADC or any other data source with slowly or rarely varying information. It is fed into a separate slow collector process typically at a 1 Hz update rate.

Every data collected from the fast channels is synchronized to a macro pulse or shot. This is done by providing an event identification number, distributed via the timing system and attached to the corresponding ADC server data. Slow data is collected by a central slow collector process and then sorted by timestamp if there's no event id available. Eventually the slow collector tags the slow events correctly with the corresponding event id. Thus, any data for a given macro pulse and its individual bunches can be easily correlated with each other, even among slow and fast data, via the event id.

Centralized Shared Memory

A *buffer manager* [3] - using up to 32 Gbyte of shared memory - keeps a buffer of synchronized data for each event id. The shared memory has a slot-like structure where fast and slow collector can write the various front-end server data they received. All the data blocks, belonging to a given event id, are kept in shared memory until the full event record is assembled, then either holds it for subscribed clients, or sends it off to the distributor and event writer processes for writing the raw data files. The fast and slow collector fill up the shared memory slots, whenever receiving data from the front ends. Any clients, using the buffer manager interface, can read event data from the shared memory, as well as they can write and add additional data to the existing event record before it will be processed further.

This feature enables so-called *middle layer servers*, which can access data not only directly from the front-ends but from shared memory, process it e.g. do computations, and put back the result into memory for storing it together with the original event data. It is furthermore available for middle layer applications and can be utilized for slow, software-implemented, feedback loops and other monitoring purposes, requiring more complicated calculations.

Scalable and Extensible Architecture

To accommodate the rapidly changing nature of an R&D machine as well as to allow for frequent changes of the beam line experiments and their equipment, the DAQ was designed to be extensible and scalable. It is possible, to run several instances of a DAQ including data collectors, buffer manager, distributors and middle layer servers either concurrently on one node or on separate machines.

Data streams can be used for separating experiment data from machine data or from any other experiments and even device groups. Streams will produce individual sets of raw data files containing only the configured subset of the available data.

A HEP-style run control and configuration database (ORACLE RDB) is used for configuring DAQ processes and any DAQ-related front-end device parameters. This allows for dedicated and individual configurations of DAQ instances. Each instance has its own run control and can thus be operated independently.

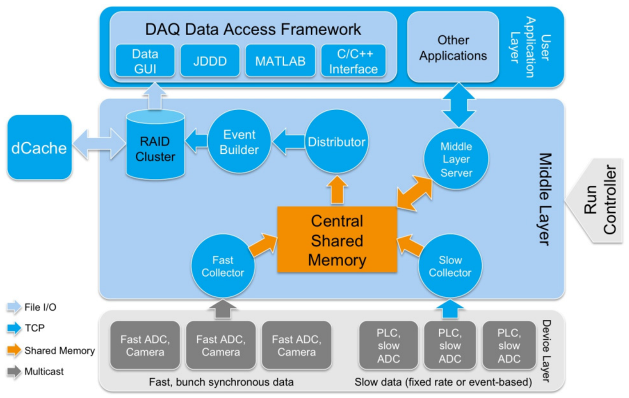

Figure 1: FLASH DAQ data flow.

EVOLUTION

The FLASH facility is now operating since August 2005. Since day one a basic DAQ implementation with two streams for the linear accelerator and one for a user experiment were present and running. Meanwhile the overall system has been significantly enhanced and upgraded. A few of the experiences will be highlighted in the following.

Evolution - Data

The FLASH DAQ produces currently about 1 Tbyte per day. This data is being written to a large area on the storage node and kept for at least 2 to 4 weeks. It migrates slowly off disk if there are not any requests by the accelerator physicists or experts to keep dedicated runs. Being *kept* on disk however and routinely written to tape via dCache, is data from the photon beam line experiments, photon diagnostics data, specific data from accelerator experiments - e.g. the electro-optic sampling device – and data for R&D accelerator studies.

At the moment the rate of 1 Tbyte per day can be handled fine, even after the upgrade to sFLASH, the data production rate will be of the same order. To be prepared for storing larger data amounts for some time as well as to allow for scalability and redundancy, the original DAQ storage cluster has been recently upgraded. Essentially the two storage nodes have been doubled and provide now a total of 60 Tbytes free disk space for the migration pool alone.

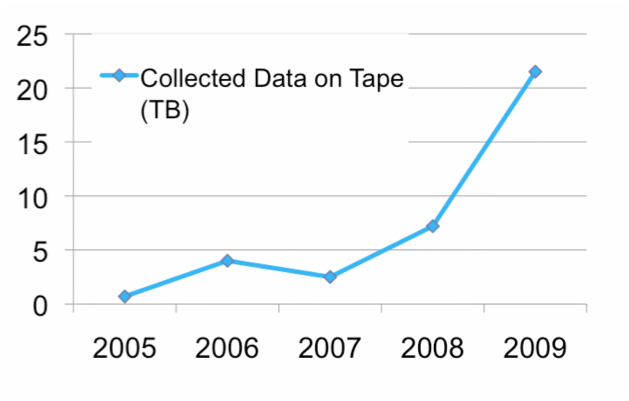

Figure 2: Data collected on tape since 2005 in TBytes.

Evolution - Scalability and Extensibility

Not only the storage node hardware has been upgraded, but also the worker nodes are doubled now. This gives some redundancy, because due to its design, it is possible to run the main and other DAQ instances on any other node within the FLASH cluster. Both worker nodes are configured, so that either one could run the main DAQ system. This feature came in handy this year, after CPU units on the original machine failed, and the new node had to take over.

Furthermore, it is relatively simple, to *add* another DAQ instance by using a script for cloning. One has still to set up the run configuration control and adjust run parameters via a Java-based GUI, but that is basically all to it. This had to be done multiple times since the original layout. Mostly beam experiments started to utilize new beam line stations, but also R&D work required some dedicated instances. In total, FLASH runs four instances now.

As another measure for redundancy, we introduced a backup DAQ instance for photon diagnostics information. These are crucial for user and their analysis so we installed another DAQ, running independently from the main instance and records the same photon diagnostics subset data, writes it to disk and the to dCache.

Because of the overall R&D nature, lots of devices have been modified or added to the DAQ as shown in Figure 3. For known and supported hardware channels can be added quickly to existing run configuration, its parameters adjusted, and the DAQ restarted within in minutes.

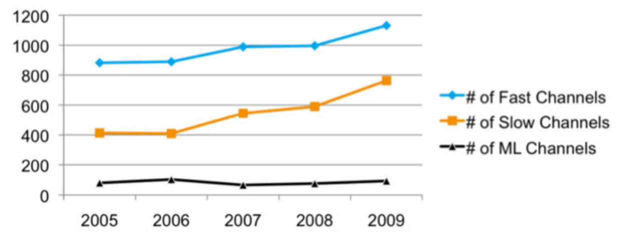

Figure 3: Number of fast (blue), slow (orange) and channels produced by middle layer servers (grey) since 2005.

EXPERIENCES

Tools and Applications = Acceptance?

A crucial question whether a data acquisition system and its framework would be accepted by the different user communities, was, whether sufficient tools and applications were available at start-up time and during the course of operations.

Clearly, the heterogeneous environment consisting of operators, accelerator physicists, experts and photon beam experiment users has been quite a challenge. Accelerator physicists have been interested since start up, but have not explored the full potential, yet. They want either ready-to-go applications or a known framework to write some (MATLAB) on their own.

On the other hand, FLASH just recently performed three sets of ILC R&D runs, the last one with long pulses and high-beam current, which did utilize the DAQ fully. 18.4 Tbyte of data for these ILC studies had been recorded.

The other big community showed most interest in using the DAQ is the photon science one. Their quite different work habits and methodology match more the HEP-style design of the FLASH DAQ. The initial attempt to use ROOT for data browsing and as an analysis framework, gained some traction within this community, but eventually that approach was dropped. It did lack performance for I/O and needed still a lot of work to set up standard tools in ROOT.

Initially an easy-to-use data browser for identifying relevant data was missing. Meanwhile Java-based tools, based on DOOCS have been created. A newly set up data server cluster provides access to all data on disk via TCP. Interfaces to MATLAB, JDDD, Java-based browser and C/C++ libraries are available.

Experience - File Format

When designing the DAQ, the initial file format was based on ROOT: it promised good compression (GZIP) and a highly flexible C++ analysis framework in HEP-style. However, when using tree-based internal structure with many different channel types, a poor performance in writing files was observed. Event writer processes could not write faster than 8 Mbytes/s to disk.

After investigating other existing solutions, we ended up creating our own raw data format, based on the ZLIB compression algorithm (or alternatively LZO). This shows rates at 58 Mbytes/s via NFS. Reading an event takes 200 μs up to 700 μs.

Experience - Middle Layer Servers

Middle layer servers became crucial for operation over time. They are used for slow feedback purposes and other applications. One can plug in any MATLAB routine or C/C++ code for computations, configure necessary input parameters (via GUI) and output parameters. Examples of middle layer servers, without FLASH would not be fully operable anymore, are: energy measurement, photon wavelength and photon energy measurement, gas monitor detector, LLRF server for high-level LLRF data processing, a quench detection server and an orbit server. Currently a lot of work is spent on enhancing the interfaces between DOOCS server and the buffer manager to improve responsiveness and robustness.

CONCLUSION

The design of the FLASH DAQ has shown to be a powerful concept and a proven architecture in general. Running reliably since 2005 for accelerator and photon science community and its experiments, it has become crucial for operations. Applications turned out to be an important factor and dependent on the community work habits to be accepted as useful tool. More applications are needed, especially, those implementing automated procedures.

Most recently the FLASH DAQ has been used with great success for ILC R&D work in 2008 and 2009 – about 18 Tbyte of data have been recorded to support the analysis. Thus, the basic concepts seem to be right, though tools and data storage need more evaluation especially in the view of the upcoming European XFEL project.

REFERENCES

[1] G. Grygiel, O. Hensler, K. Rehlich, “DOOCS: A Distributed Object-Oriented Control System on PCs and Workstations”, PCaPAC ‘06, Hamburg, Germany, 2006, see http://doocs.desy.de.
[2] A.Agababyan, et al, “Multi-Processor Based Fast Data Acquisition for a Free Electron Laser and Experiments”, IEEE Transactions on Nuclear Science, Vol. 55, No.1, February 2007, p. 256.
[3] V. Rybnikov et al, “Buffer Manager Implementation for the FLASH Data Acquisition System”, Proceedings of PCaPAC 2008, Ljubljana, Slovenia, 2008, p. 102.