FLASH DAQ DATA MANAGEMENT AND ACCESS TOOLS

* V.Rybnikov, V.Kocharyan, K.Rehlich, E.Sombrowski, T.Wilksen

Abstract

The Free Electron Laser in Hamburg (FLASH)[1] at DESY is a user facility for the photon science community. It produces laser light of short wavelengths from the extreme ultraviolet down to soft X-rays. To study, monitor and document the machine performance and parameters and also to collect the results of the experiment measurements, a fast data acquisition (DAQ) system is being used. Having above 1000 linear accelerator diagnostics channels collected by the DAQ currently results in a data rate of ~100 Mb/s. The large amount of data requires corresponding data storage and management to enable efficient data retrieval. This paper will focus on the data paths, storage and bookkeeping. A number of tools provided for the users to work with DAQ data will be described. The current status of the achieved performance in the data storage and retrieval will be covered as well.

INTRODUCTION

The FLASH DAQ [2] system was launched in summer 2004. Its main tasks are: collecting LINAC beam relevant data in real time, providing the data to feed-back and monitoring tools as well as storing it for an offline analysis. The DAQ system is also used by FLASH user experiments to store their data together with information coming from LINAC. This allows easy correlations between the experiment measurements and the LINAC state. A set of tools is provided for data visualization and analysis.

DATAFLOW

The dataflow in the FLASH DAQ and all involved components are shown in Fig. 1. There are two types of data collected by the DAQ. Fast data include channels with beam related information (beam position monitors, etc.) and currently collected with the shot repetition rate of 10 Hz. All other channels considered as slow (magnet currents, etc) and collected with the maximum rate of 1 Hz. The data is collected by fast (FC) and slow collectors (SC) correspondingly via Ethernet. The collectors put data to the Buffer Manager (BM) [3] for online access. Distributors (DS) read data from the BM according to the stream descriptions provided by the Run Control during the DAQ configuration procedure. The data streams are pushed to the Event Builder (EVB) and further to the Writers (WR). The latter writes data to files on a local disk. The files from the local disk are copied to a huge RAID array and accessible via NFS for the public. The experiments data is usually copied to tape for the permanent storage.

*Deutsches Elektronen-Synchrotron, DESY, Hamburg, Germany

DATA MANAGEMENT

The DAQ data management components control the data flow and guarantee all required data is written to data files and to the tape if required. The components keep track of the written data in order to assure fast data access. The rest of the paper will be devoted to the description of those components.

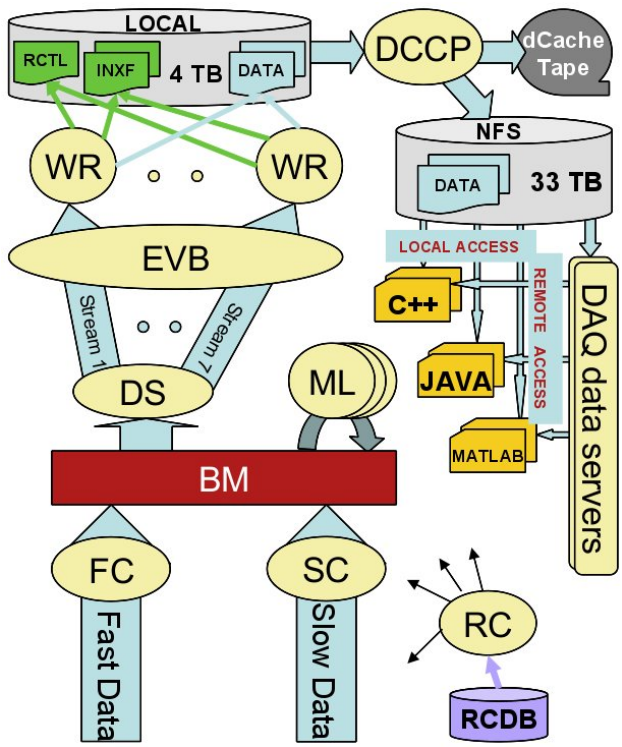

Figure 1: FLASH DAQ dataflow and access tools

Fast Channel Data

The fast channels (~ 700 channels) provide the most part of the data volume (~99.5%). It means that the reduction of the total data amount strongly depends on the configuration of the front-end DAQ senders. The front-end configuration is performed by the Run Control process during the DAQ configuration. The RC is capable to set every parameter for the spectra that are usually sent by the front-end (e.g. start, increment, length). For that the RC has a group of run parameters when changing one of them changes a spectrum parameter in a group of channels (e.g. the same device types). The different sets of run parameters are stored in the Run Modes of the Run Control data base [4]. Every Run Mode corresponds to a

FLASH operations mode. In this way one can control the total amount of written data.

Data produced by Middle Servers (MS) belongs to the fast data. Their configuration is performed by the RC too, and therefore defined by the Run Modes.

Slow Channel Data

The amount of slow channel data is controlled by assigning the channels to two types of slow events: update event and/or environment event. The data for update events is put to the BM periodically (currently every 15 seconds). The channel data for environment events is put to the BM only on its value change. Writing the environment channels can be also controlled by filters (absolute or relative limits for the value difference).

Data Streams

The FLASH DAQ is currently writing 7 data streams simultaneously. Every data stream consists of a set of collected channels. The list of channels for a stream is defined by the experiment using the stream. The largest LINAC stream contains all data collected by the DAQ. The LINAC data can be used by other experiments in case some additional channels are required for their analysis. The stream separation is done by the DS. It receives stream descriptions from the RC during the configuration.

Writers, Run Catalogue and Index Files

Currently EVB is acting as a gateway between distributors and writers. In future one could use it as filter for data streams to reduce the data volume, to generate statistics, etc.

Writer processes are responsible for dumping the data streams into files. The writers keep track of created files by means of a Run Catalogue (RCTL). For every run and stream a set of index files (INXF) is created by the writers. The RCTL is a binary file that contains the start and the stop time for every run and the number of written files. The index files contain the information about every written file. It includes time stamp and event IDs of the first and the last event in the file and the number of events for every event type. The RCTL and INXF allow to find the list of file names for a certain time period and experiment.

Permanent Data Storage

The files written by WRs are shipped from the local disk to the tape storage by the dCache [5] copy process (DCCP). DCCP keeps track of all taped files in a DCCP catalogue.

FLASH DAQ Data Files

We are using a custom designed file format for the data storage. The format is highly optimized for fast data access. The fast access is achieved by writing the data for one channel in a continuous data block (basket). Three steps are usually required to read a channel data:

- Read reference tables and find out the data basket offset
- Read the data basket
- Decompress data if required

Depending on the data type the channel data can be written into the file with or without compression. Two algorithms for compression are supported: ZLIB [6] and LZO [7]. The second one is used in case of CPU power limitations.

DAQ data files are self describing. One can get information about the list of stored channels with their descriptions as well as the number of entries for every channel without accessing the data itself.

DATA ACCESS TOOLS

To be able to work with the DAQ files one needs tools to extract the required data. A set of tools has been developed, providing two different access methods: directly from files (local access) and by means of DAQ data servers (remote access, see Fig.1). In the second case the user software receives the required data from the DAQ data servers running on dedicated computers. The requirements to the data access tools are strongly dependent on the user's task. We have concentrated on the general purpose visualization tools and the libraries that could be used by the experts to make their own processing programs corresponding to their wishes. We have developed libraries and tools for thee environments used at FLASH: C++, Java, MATLAB [8]. Platform supported are Solaris (SPARC), Linux (Debian and Ubuntu) as well as Mac OS X.

C++ BASED TOOLS

A set of classes has been developed to access DAQ data from files. In order to start the data extraction one needs to provide a "data request" containing the time period (start, stop, or a run number), list of channels to extract and the experiment name. One can set all those parameters either by means of the corresponding methods of classes or by providing the name of a XML file containing all required information. Once the request is defined, a method to start the data extraction is to be invoked. Due to multithreading design one can get the required data simultaneously with ongoing data extraction.

Based on the described libraries a data processing framework has been developed. It takes care of the data extraction. The user is provided with 4 routines: user_help(), user_init(), user_loop() and user_end(). The routines can be rewritten by the user, recompiled and re-linked with the framework. The loop routine is called to provide the user code with the channels data in the sequence as it was collected by the DAQ. The init and end routines are called once for user code initialization and finalization correspondingly. The FLASH accelerator and photon experiment groups are using this framework since quite some time to their satisfaction.

JAVA BASED TOOLS

A library (JDAQ) has been developed in Java to provide DAQ data access for Java based applications. The library offers both local and remote DAQ data access. The library contains the same classes as the C++ ones. The same XML configurations can be used for both environments. A few Java GUIs exploding JDAQ have been developed: FLASH DAQ data GUI, FLASH DAQ data Converter, JDDD [9] expert panels.

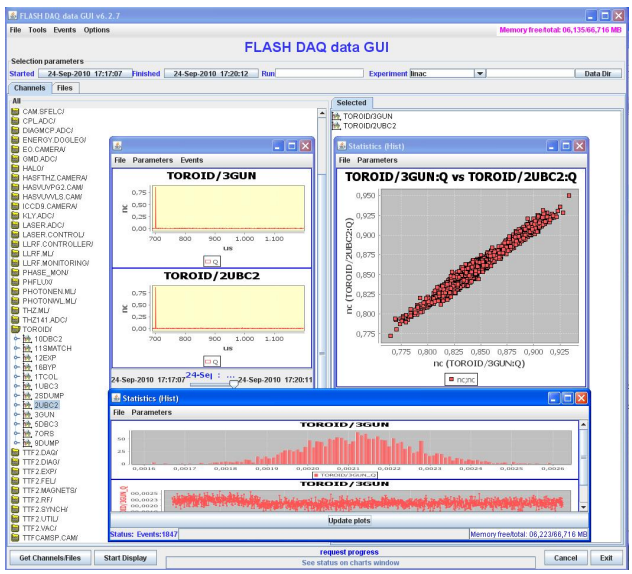

Figure 2: FLASH DAQ data GUI

The FLASH DAQ data GUI (see Fig. 2) is a general visualization tool. It makes use of JFreeChart [10] to draw waveform, histories and histogram plots. The GUI allows watching a set of channels signals as they were during every shot in the linac. One can create histograms and histories for every selected channel as well as the correlation plot for any pair of channels. The GUI can be used for local and remote data access.

The FLASH DAQ data converter dumps the DAQ data to ASCII files. The GUI has a convenient interface to plug-in other converters for producing other data formats.

JDDD is becoming the new display tool for FLASH operators and experts. A few additional components based on JDAQ have been written. They allow drawing stored DAQ data along with online data read from the FLASH control system. Because of that new capability of JDDD one can easily build panels for the experts to analyze the behaviour of their setups in the past.

The LLRF expert coupler interlock panel is a good example of that approach. The expert selects an interlock event shown in the DOOCS [11] history of the beam interlock system. On selection an event a request to DAQ data servers to extract data for a set of channels is sent for 5 seconds period before and after the event. On receiving the requested data the wave forms are plotted in the same plots where the current online waveforms are drawn. The expert can shot by shot go through 10 second period of DAQ data and compare the channels with the online ones. In this way it makes it easy to find out the source of the interlock for the selected interlock event.

MATLAB TOOLS

Based on the C++ classes external MEX functions have been developed to provide access to the DAQ data from within MATLAB. The DAQ data request can be set either by setting an array of strings inside of the MATLAB script or via an XML file with the same format as for C++ and JDAQ libraries. The MEX functions extract the requested data and convert it into MATLAB structures that can be read by MATLAB scripts and analysis code.

PERFORMANCE

The FLASH DAQ currently collects all required beam related channels with the rate of 8000 bunch/s (800 bunches at 10 Hz repetition rate).

The measurement of the data extraction time shows that for modern workstations exploiting fast multi-core processors it mostly depends on disks performance and network bandwidth. In our environment we measure 0.1-0.2 ms/event for reading one spectra channel (2000 floats).

PLANS

We continue to improve our data access tools trying to satisfy our users' requirements. E.g. implementing pre-processing of data by the DAQ data servers could drastically reduce the amount of raw data currently used by the clients and speed up the final data processing.

REFERENCES

[1] http://flash.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 2008.

[3] V. Rybnikov et al., "A Buffer Manager Implementation for the FLASH Data Acquisition System", PCaPAC 2008, Ljubljana, Slovenia, October 2008

[4] G.Dimitrov, "Application of Oracle Database for TTF DAQ System", PCaPAC 2005, Hayama, Japan, March 2005

[5] http://www.dcache.org/

[6] http://www.zlib.net/

[7] http://www.oberhumer.com/opensource/lzo/

[8] http://www.mathworks.com/products/matlab/

[9] http://jddd.desy.de

[10] http://www.jfree.org/jfreechart/

[11] http://doocs.desy.de