DATA ACQUISITION SYSTEM FOR A VUV-FEL LINAC

*A.Agababyan, G.Grygiel, B.Fominykh, O.Hensler, R.Kammering, K.Rehlich, V.Rybnikov,
**L.Petrosyan, ***A.Asova, G.Dimitrov, G.Trowitzsch, M.Winde, ****T.Wilksen.

Abstract

For the VUV-FEL (Vacuum-Ultraviolet Free Electron Laser) at DESY a 1 GeV superconducting LINAC [1] is used and presently being commissioned. The control system of the LINAC consists of 900 fast and 500 slow data channels producing up to 100 MByte/s of data depending on the LINAC operation mode. To study, control and archive the machine status and performance, a data acquisition system (DAQ) has been developed which allows storing machine parameters together with user experiment data. The system will provide the ability to correlate easily the accelerator status with experimental data on a bunch-by-bunch basis. The paper gives an overview on the architecture of the DAQ system and describes its main components in more detail.

INTRODUCTION

Traditionally data acquisition systems are used in High Energy Physics experiments (HEP) where large amounts of detector data is sequentially reduced in size on its way from the particle detectors to the archiving tape. The results stored on a permanent medium allow reconstructing the physics processes during the experiment.

The experience of the Tesla Test Facility phase 1 setup (TTF1)[2] has shown that for a very complex system like the VUV-FEL LINAC one has to be able to collect all beam relevant data in time to use them for monitoring and automatic feed-back systems to achieve the best machine performance.

The DAQ system for the VUV-FEL LINAC is dedicated to the following tasks: 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 will also enable storing user experiments data in parallel streams for making further correlations between the experiment measurements and the LINAC state. Tools for analyzing the stored data both for local and remote users will be provided.

DAQ LAYOUT

Since the DAQ system was meant for collecting not only the LINAC but also for user experiments data, its architecture has to be very flexible and scalable to integrate 'short live time' experiments as well as any future modifications of the LINAC. Taking into account that the VUV-FEL LINAC will be used by experiments from off-site institutions the DAQ system should provide means for remote LINAC operations.

For reading out the various front end devices and user experiments data mostly the Distributed Object Oriented Control System (DOOCS) [3] is used as well as for inter-process communication, control and configuration of higher level servers.

The layout of the DAQ system is shown in Fig.1. We consider channels with beam related information as fast (beam position monitors, etc.) and all other as slow (magnet currents, etc). The data is collected by fast (FC) and slow collectors (SC) correspondingly via a network.

Both types of collectors are running on a server computer and use of custom written Buffer Manager (BM) API for temporary data storage. Middle Layer server programs (MS) take the collected data and process it for special purpose operator monitors and feed-back subsystems.

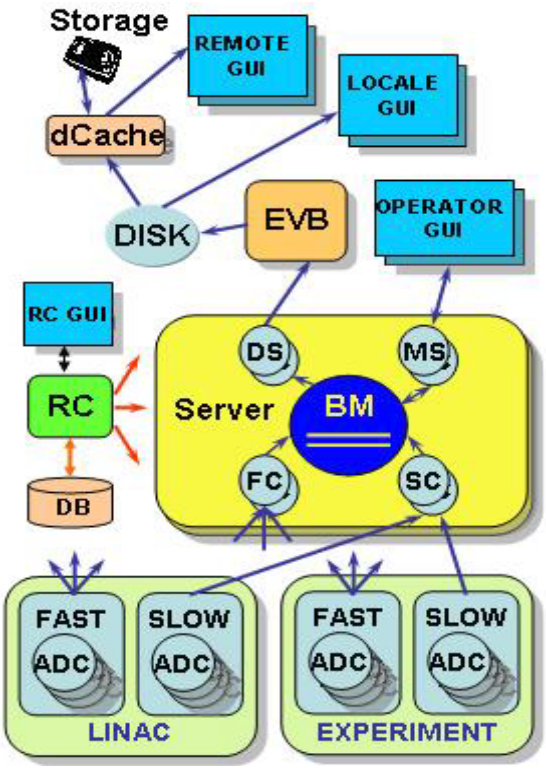

Figure 1: DAQ layout.

Data distributor programs (DS) push the data to the Event Builder (EVB) which in turn stores the data on a local disk both for online monitoring (Locale GUI) and for remote analysis after re-writing the data to a disk cache (dCache) [4]. The later is also used as an intermediate storage before writing the data to tape.

The Run Control process (RC) is in charge of the configuration and monitoring the whole system. The RC

*Deutsches Elektronen-Synchrotron, DESY, Hamburg, Germany
**YerPhi, Yerevan Physics Institute, Armenia, currently at DESY
*** Deutsches Elektronen-Synchrotron, DESY, Zeuthen, Germany
****LEPP Cornell University, Ithaca, NY, USA

is storing all DAQ parameters in a Run Control Database implemented as a Oracle Data Base [5].

A detailed description of all above mentioned DAQ components is given below.

Front-end fast channel data collection

Front-end fast data channels are distributed among 40 VME crates. One VME crate can carry up to 8 fast 8-channel ADC boards. ADC channels are usually assigned to different device types (depending on the crate's physical position with respect to the LINAC). Channels with the same device type are typically controlled by one DOOCS ADC server.

All front-end ADC channels are driven by the central timing system used for the LINAC synchronisation.

Front-end ADC servers perform the following tasks: copying data from ADCs to the internal memory according to the bunch type (single, short, interrupted, full), converting raw ADC data into physical units, calculating mean values, filtering data configured by the Run Control, packing the prepared data into a "server block" and sending it via a multicast. Usage of multicast transfers allows us to collect once sent data by many Fast Collectors that can run on different computers at the same time.

Configured by the RC Fast Collectors subscribe to corresponding multicast packages on the network. They assemble whole events from the received server blocks sent by ADC servers.

The DAQ system considers as many fast event types as required for data separation according to its origin source (beam diagnostics, beam profile measurements, etc.).

Synchronization of server blocks is implemented via a time stamp obtained by every ADC server as UNIX time. Running Network Time Protocol (NTP) [6] daemons on every front-end computer guarantees that we can distinguish data belonging to subsequent micro-pulses of the LINAC for the maximum repetition rate of 10 Hz.

On collection an event FC stores it in the Shared Memory (SHM) [7] by means of BM.

Slow channel data delivery

We consider the maximum update rate of slow channels not faster then 1 Hz. The multi-threaded design of the Slow Collector (using POSIX threads [8]) allows collecting data from many slow channels simultaneously. The system foresees having several SC running either on the same computer or on different ones. DOOCS is deployed by SC for the data collection.

SC creates events at maximum rate of 1Hz. Data from a particular channel is included into the new event only if the new measured value exceeds some limits preset for this channel by the RC.

We consider several types of slow events. They are Begin of Run (BOR), Update (UPD), End of Run (EOR) and Environment (ENV) events.

BOR and EOR are generated only once for a LINAC run and represent slow channel data that is meant to be constant for a particular LINAC run mode.

UPD events contain data from the channels that we need periodically. The time period of UPD events is set by the RC and can be changed during the run if required.

ENV event is the basic slow event type containing slow channel data. The maximum rate of ENV events is 1 Hz. SC includes a slow channel data corresponding to the procedure described above.

We have one additional event type called Heart Beat (HB). This event type contains the information for those processes that can connect to the DAQ system in the middle of the run but require the complete information about it.

Middle Layer servers

All processes that need events from the BM have to subscribe to the events of interest. The subscription is done on the event type basis. Once there is a new event in the BM collected either by FC or SC all subscribed processes are notified to let them process the event.

Every ML server usually solves one task of either monitoring or controlling the LINAC or making some measurements. As middle layer servers are currently implemented Orbit Server, Energy Server, Charge server and Wire Scanner Control server.

Every ML server can also produce its own events of a special type containing the results of its calculations that can be used in further online or offline analysis.

Distributor server

Distributor processes fulfil two main tasks: creating data streams, pushing them to EVB. We consider that not all data collected in BM can be of interest for further analysis. Moreover we limit ourselves with 4 PB/Year of data on tape. On the other hand for the analysis convenience the relevant data should be stored in the same file. Therefore the DS makes selection of events at two levels: on event type level and on data channel level, according to the configuration from RC. Once an event with pre-selected data channels is produced by DS it's sent to EVB via TCP/IP [9] connection.

The DAQ foresees up to 32 data streams produced simultaneously.

Event Builder

The EVB receives data from DSs by means of dedicated Data Receiver threads. The number of those threads is configured by the RC and corresponds to the number of DSs (streams) in the system. For every data stream EVB can also have some additional filtering of data as well as creating additional streams if required.

We have selected ROOT [10] system for writing output files as the main format. Although some additional plug in modules can by applied to streams by EVB to produce files in other formats (e.g. ASCII or MatLab [11] compatible file format) if required.

EVB writes data files to a local disk in the corresponding directories according to the experiment name, stream name and some other parameters obtained from the RC. Along with writing data files some

additional files a stored on the local disk as well e.g. Run Catalogue, streams descriptions and index files. They contain the information required for offline analysis. With their help one can find out the list of channels in the selected stream, names of files to analyze for a certain time period.

Permanent Data Storage

The ROOT files written by EVB are shipped from the local disk to the tape storage. A future worker task possibly implemented as a DOOCS server will manage these files and feed them into the disk cache system to be used for data file staging to and from the tape. The dCache has several GRID [12] interfaces and therefore provides an easy access for experts to retrieve the event files remotely.

Graphics User Interfaces

There are various GUIs used at different levels of the DAQ, fulfilling different tasks.

Operators GUI implemented as DDD and DOOCS GUIs [13] are mostly meant to monitor the LINAC status. They can also be used for launching some measurements that are carried out from time to time if required (e.g. Wire Scanner for beam profile measurements [14]). Operators GUI also allow changing LINAC parameters within the same run mode (see Fig.2).

Locale GUIs allow to process data from ROOT files. Complex and- time consuming data analysis tasks are usually handled by Locale GUIs. This is the reason why they get data from ROOT files and not from BM.

The DAQ Browser [15] is the main Locale GUI tool and is based on Qt [16] and ROOT. This tool is able to analyze also very complex correlations and dependencies on pre-selected data channels for a chosen time period.

Remote GUIs will be used by physicists and LINAC subsystems experts for offline analysis. The choice of GUIs will depend on the experts but standard tools like MATLAB as well as ROOT will be supported. Ability to work with ROOT files and accessing them via one of DESY supported interfaces (e.g. GRID) are the only requirements to the Remote GUIs.

Run Control

The Run Control process is responsible for configuration of both DAQ and LINAC parameters according to the LINAC run mode. Possible LINAC modes are shown in Fig.2.

RC uses an Oracle data base to store all required configuration parameters. The whole information is distributed among several tables. The relations between fields of diverse tables are heavily used to avoid information duplication.

The RC data base [17] allows preparing the configuration parameters for a new LINAC run as the following. The first step is the selection of the LINAC mode. Every LINAC mode defines a list of hardware and software entities that have to be configured via a set of their parameters. Every LINAC mode can contain several versions of those parameter sets, meaning the same beam path but with different properties. The RC DB allows having some run parameters that can be changed by operators before starting the run. One run parameter is usually associated with a group of hardware or software properties. This gives a very flexible way to change a lot of parameters in the system by setting once the corresponding run parameter via the RC.

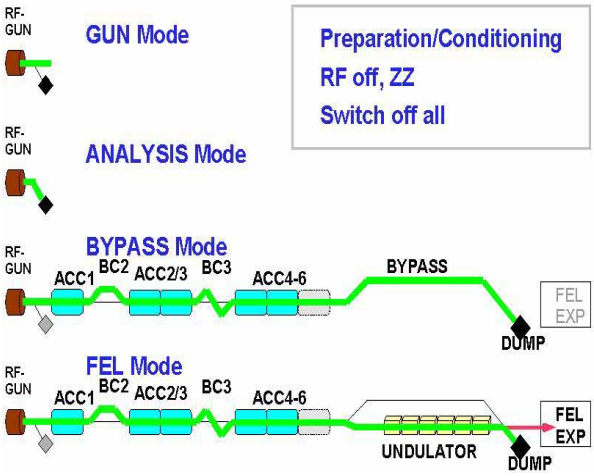

Figure 2: LINAC run modes.

To prepare the system for a LINAC run the RC takes the following steps: chooses the run mode choice, selects the run mode version and changes some run parameters if required. The RC also provides the possibility of more detailed software and hardware selection for the DAQ usage. This is important if some subsystems have problems to run properly. All those settings are done by means of the RC GUI. Once a new run mode is set up, the RC configures all LINAC and DAQ processes via the distribution of the parameters. After that the RC makes use of two DAQ processes (not shown in Fig.1) that drives the corresponding Finite State Machines (FSM) [18] of the LINAC and DAQ processes to bring them to the final 'RUN" state. In this state the LINAC produces the beam according to the mode and the DAQ collects data according to its configuration.

Once a run is set up the operators can tune the machine parameters with operator GUI in order to achieve the best performance. On reaching this point all machine settings can be stored by RC in the DB as a new run mode version.

CURRENT STATUS

The DAQ system has been setup in the debugging mode since the end of the last year. The system is not complete yet, but does allow writing ROOT data files on a local disk.

A SUN Work Station (Solaris OS) carrying 16 SPARC [19] processors running with 1,2GHz, and 32 GB of working memory is employed as the Server (see Fig.1). The machine contains two Gigabyte Ethernet interfaces giving us in total ~200MB/s network bandwidth.

Our tests have shown that the modern network hardware provides very reliable data delivery even under high network load and UDP [8] protocol employment when using multicasts. The fraction of events with incomplete data has been measured at the level of less then 0,3%. The network bandwidth occupied by the collected data made up to 25MB/s. The CPU consumption by FC was as low as 20%. These measurement results make us optimistic to achieve our final goal for the DAQ system to be able to sustain ~100MB/s of data through BM.

The DAQ system architecture easily allows having at least one additional BM and all involved processes (FS, SC, MS, DS) running in parallel with the main DAQ system collecting at least the same data. This is important for the commissioning and further development of the system once LINAC is controlled by DAQ.

PLANS

We plan to start the DAQ commissioning in spring this year. The commissioning will be done it two steps. At the beginning the DAQ will be providing all MS with data from BM. RC will configure only DAQ processes. The main goals of the first stage are to make the LINAC operators get used to the DAQ system and to gain experience of its exploitation.

Once DAQ is set up and runs smoothly we will switch on its feature to configure LINAC parameters as well as the mode versions creation.

CONCLUSIONS

Along with solving the task listed in the introduction, the deployment of the full featured DAQ system will provide us with:

- better understanding, operating and maintenance of the LINAC
- easier way of error handling: finding reasons of faults, reliability improvement
- operation optimization, finding best parameters
- correlation of complex machine parts

The VUV-FEL LINAC DAQ system is the result of successful collaborative work of three institutes:

- Deutsches Elektronen-Synchrotron, DESY, Hamburg, Germany
- Deutsches Elektronen-Synchrotron, DESY, Zeuthen, Germany
- LEPP Cornell University, Ithaca, NY, USA

REFERENCES

[1] A VUV Free Electron Laser at the TESLA Test Facility at DESY - Conceptual Design Report – DESY.

[2] TESLA Test Facility Linac – Design Report, Mar. 1995, DESY Print, TESLA 95-02.

[3] G.Grygiel, O.Hensler, K.Rehlich, “DOOCS: a Distributed Object Oriented Control System on PC’s and Workstations”, PCaPAC96, DESY, Hamburg, October 1996.

[4] M.Ernst, P.Fuhrmann, M.Gasthuber, T.Mkrtchyan, C.Waldman, “dCache, a distributed storage data caching system”, CHEP 2001, Beijing, http://www.ihep.ac.cn/~chep01/

[5] I. Stevenson, D.Ensor, “Oracle design”, O’REILLY, 1st Edition March 1997, ISBN:1-56592-268-9

[6] Computer Time Synchronization, A Beginner's Guide to Network Time Protocol (NTP), http://geodsoft.com/howto/timesync/

[7] Richard Mc Dougall, Jim Mauro ”SOLARIS internals”, 2000, ISBN 0-13-022496-0, http://www.solarisinternals.com/si/10.thebook.php

[8] David R. Butenhof, “Programming with POSIX Threads”, Addison-Wesley, 1997

[9] W. Richard Stevens , “TCP/IP Illustrated, Volume 1”, The Protocols, Addison-Wesley, 1994, ISBN 0-201-63346-9

[10] R.Brune, F.Rademakers, “ROOT - An Object Oriented Data Analysis Framework”, AIHENP conference, Laussanne, Sep. 1996, p. 81.

[11] T.A.Davis, K.Sigmon, “MATLAB Primer”, CRC Press, 7-th edition, 2004, ISBN: 1584885238

[12] I. Foster and C. Kesselman (eds.) ”The Grid: Blueprint for a new Computing Infrastructure”, Morgan Kaufmann Publishers, 1998, ISBN 1-55860-475-8

[13] http://tesla.desy.de/doocs/ddd/ddd_intro.html

[14] E.Sombrowski, “Wire Scanner Control and Display Software”, these proceedings

[15] K.Abrahamyan, G.Asova, etl. “Automic Procedures as Generated Analysis Tool”, CHEP’04, Sep.27-Oct.1, 2004, Interlaken, Switzerland

[16] M.K.Dalheimer, ”Programming with Qt”, O’REILLY, 2nd Edition March 2002 ISBN: 0-596-00064-2

[17] G.Dimitrov, “Application of Oracle Database for TTF DAQ System”, these proceedings

[18] Finite state machine, http://en.wikipedia.org/wiki/Finite_state_machine

[19] http://www.sparc.com/