A data quality monitoring software framework for the BESIII experiment*

HU Ji-Feng(胡继峰)[1;1)] ZHENG Yang-Heng(郑阳恒)[1]
SUN Xiao-Dong(孙晓东)[2;2)] JI Xiao-Bin(季晓斌)[2]

[1] Graduate University of Chinese Academy of Sciences, Beijing 100049, China
[2] Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China

Abstract: Data quality monitoring (DQM) plays an important role in data taking at the BESIII experiments. DQM is used to monitor detector status and data quality. A DQM framework (DQMF) has been developed to make it possible to reuse the BESIII offline reconstruction system in the online environment. In this framework, the DQMF can also simulate a virtual data taking environment, transfer events to the event display, publish histograms to a histogram presenter in a fixed interval, and dump histograms into a ROOT file. The DQMF has been stably running throughout BESIII data taking.

Key words: BESIII, DQM, BOSS, emulator

PACS: 29.40.Mc, 29.30.Hs **DOI:** 10.1088/1674-1137/36/1/010

1 Introduction

The upgraded BEPC II is a multi-bunch and high luminosity collider, which has achieved a peak luminosity of 5.9×10^{32} cm$^2 \cdot$ s^{-1}. The BESIII detector is the only general facility at BEPC II, and is designed to fulfill the requirements of the physics [1] and operate in the τ-charm energy region. Studies of τ-charm physics could reveal or indicate the possible presence of new physics in the low energy region. BEPCII/BESIII started operation in summer 2008, and eight high-quality papers based on the collected data have been published. Many other interesting analyses, involved in light hadron spectroscopy, charmonium physics and charm physics, have also been carried out.

During BESIII data taking, events are assembled in raw data format, and then flushed into persistent storage in the online environment. These tasks are done with the data acquisition system (DAQ) [2]. The DAQ also provides the hit-maps of sub-detectors. However, these hit-maps are insufficient to monitor data quality, so a better solution is to develop the data quality monitoring system in the common software framework, the data quality monitoring framework (DQMF). The DQMF takes advantage of the full offline reconstruction flow and physics analysis algorithms to produce histograms, which are used to monitor detector status and data quality.

Since the detector works at an event rate of about 4000 Hz, it is difficult to process every event in real-time due to limited computing power. The DQMF is required to process events as much as possible, and to run stably and automatically during data taking. The DQMF is also required to keep the two environments compatible so that new developments and updates of software packages can be added to the system smoothly. To achieve the goals, the DQMF is designed to process events in a sampling mode and with a client/server (C/S) structure to separate the two environments.

This letter focuses on the DQMF structure, the key design points, the framework implementation and other related issues.

Received 21 March 2011, Revised 19 April 2011

* Supported by Ministry of Science and Technology of China (2009CB8252000), Joint Funds of National Natural Science Foundation of China (11079008), Chinese Academy of Science (CAS) Large-Scale Scientific Facility Program, 100 Talents Program of CAS and Natural Science Foundation of China (10605030)

1) E-mail: hujifeng@gucas.ac.cn

2) E-mail: sunxd@ihep.ac.cn

2 DQM framework design

As shown in Fig. 1, the DQMF consists of several relatively independent components cooperating to meet the requirements of DQM.

DQM servers and clients are the underlying components, and they are designed with a C/S structure. In this design the DQM server offers the raw format events taken in real time, while DQM clients work in a flow similar to the BESIII offline software system (BOSS) [3]. In the flow the DQM clients receive the raw events, invoke BOSS algorithms to reconstruct the events, and invoke pre-defined physics algorithms to analyze the events, then produce histograms to show the detector performance. These histograms are displayed to shift crew in realtime at the BESIII control room with corresponding reference histograms. Beyond the realtime information, the DQMF dumps the histograms into the ROOT [4] files run by run. These monitored performance quantities, at both the basic level and the high level, include the spatial resolution of the main drift chamber (MDC) [5] wires, the time resolution of the TOF [6] counters, the energy resolution of the electro-magnetic calorimeter (EMC) [7] cells, etc., and some kinematic variables.

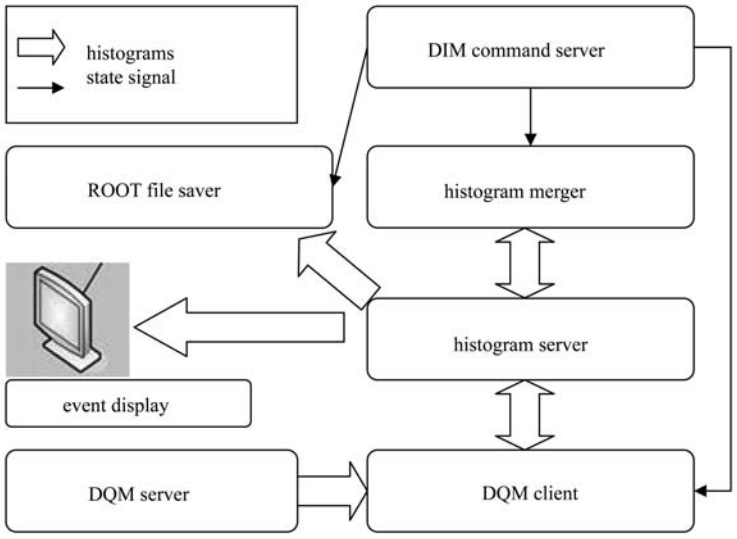

Fig. 1. Overview of the DQM framework.

The distributed information manager (DIM) [8] command server is used to automatically send the DIM commands to drive the DQMF. The commands are converted from detector states, which are synchronized with the state machine of the DAQ.

In the DQMF four DIM commands, "BeginRun", "Run", "EndRun" and "Wait", are used to describe a complete cycle of operation, which is defined as follows.

1) BeginRun: DQMF initializes each algorithm, and the histograms of the last run are dumped to the ROOT file.

2) Run: full-reconstruction algorithms and analysis algorithms are invoked to produce histograms.

3) EndRun: some algorithms fetch histograms from the histogram server for extra processing.

4) Wait: waiting for commands.

The histogram server is used to keep histograms, which are produced by the offline reconstruction, the physics analysis algorithms, and the histogram merger. The histogram server is beyond the control of DIM commands, and provides some useful histograms to a GUI presenter.

The histogram merger aims to merge those histograms from different DQM clients. The merger only communicates with the histogram server, which keeps histograms in an independent directory for each client. This design separates the DQM clients from each other. Any DQM client broken in connection with the histogram server, doesn't affect the others. The merger fetches the histograms from each directory, and histograms with the same name are merged into one histogram, which is published on the histogram sever. The merging flow is performed at a fixed interval of 30 seconds. In addition, we have an optional merging flow, which starts the merging flow only if a new updated histogram is received, which could reduce the computing costs.

The ROOT FileSaver is responsible for dumping histograms on the histogram server into a ROOT file run by run. When receiving the state-signal "BeginRun", the dumping action starts, then resets the histograms on the histogram server.

3 Implementation

The DQMF has been implemented in C++ language based on the ATLAS [9] DQMF packages and the offline environment. Each part is developed as an independent module, and explained in the following paragraphs.

3.1 Client/server communication

The C/S structure shown in Fig. 2 is adopted for event exchange to separate the online environment from the offline environment effectively, which successfully overcomes many incompatibility problems and avoids potential inconsistencies in upgrading the software packages.

The DQM server could be regarded as an events provider. It works in the online environment, uses raw data as its input, and provides events continuously. The DQM client is an event consumer. It works in the offline environment, receives events from the DQM server, and produces histograms as its output.

The DQM server is designed in object-oriented programming, and implemented as a class. The communication function between the DQM client and server is based on the socket package. The function of performing offline reconstruction and physics analysis is implemented by encapsulating BOSS. A DQM-Server.exe process is provided to offer raw events, while 20-DQMClient.exe is used to process the events.

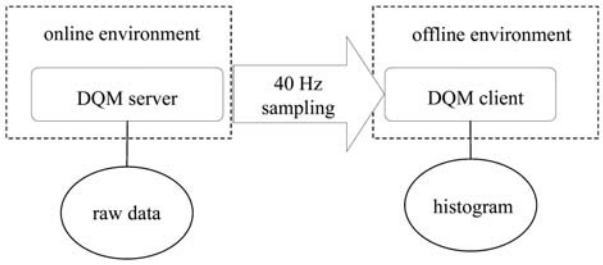

Fig. 2. DQM client/server structure.

The DQM server is in charge of sending continuous events to each DQM client, and can stably run without any extra operation from the shift crew. It is implemented with multi-threaded programming. Each thread processes a request from a DQM client, and independently offers raw data events to the client. This effectively improves the processing performance. On the same ground, the DQM client is designed with two parallel threads (a main thread and a branch thread) and a shared queue.

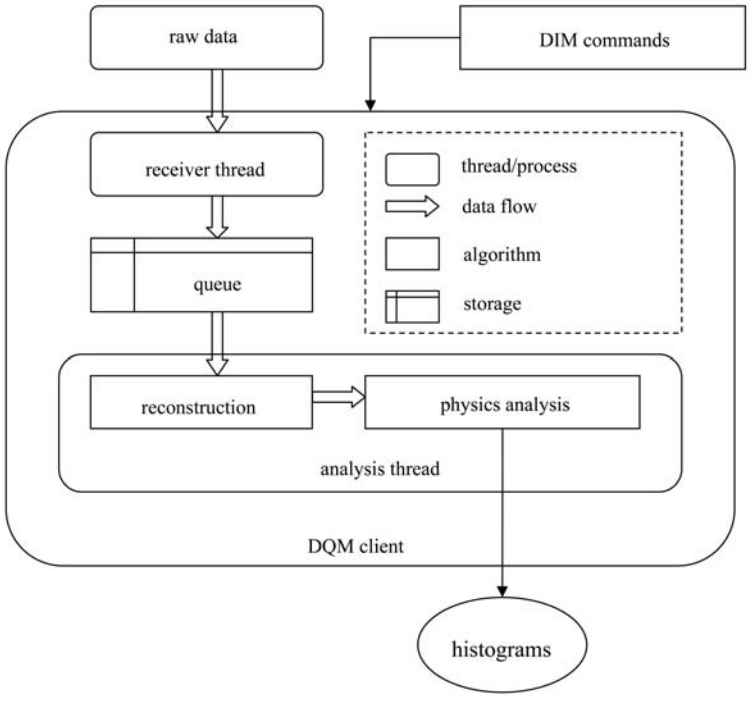

Fig. 3. The data flow in the DQM client.

3.2 The client

The DQM client shown in Fig. 3, is designed to perform full reconstruction and physics analyses to check the data quality at different levels. An event rate of greater than 40 Hz is required for processing ability. The following strategies are applied to achieve this goal. DQM clients are running on five PCs (four cores/PC), and each DQM client is designed with one event receiver thread, one analysis thread and a shared queue.

The receiver thread of the DQM client is in charge of receiving events from the DQM server via the established TCP/IP connection. To enhance the robustness, an extra thread manager is introduced to recover the broken connection. When the thread receives one event successfully, it immediately puts the event into the shared queue. If the queue is full the thread will trap, to wait until it is woken by the queue. The queue of DQM client is used to store and buffer the events.

The analysis thread of the DQM client is in charge of performing the standard processing flow for an event. When receiving the state-signal "BeginRun", the main thread is initialized. Once receiving the state-signal "Run", the main thread fetches one event from the shared queue and invokes DQM algorithms to process the event, as long as the queue is not empty. If the queue is empty, the analysis thread traps to wait until it is woken by the queue. On successfully processing an event, the many histograms defined in the different DQM algorithms can be updated and published on the histogram server. The DQM algorithms are standard BOSS algorithms, and in BOSS environments they are easily added or removed.

3.3 The other components

The DIM command server is developed based on the DIM package, which provides a distributed environment. The DIM command server listens to the state machine of data acquisition and translates the states into four state-signals, which are sent to the other parts of the DQMF. The DIM command server runs in background mode during data taking.

The histogram merger is based on the ATLAS [9] online packages, ROOT [4] and DIM command. It takes advantage of the interface of the online package to fetch specified histograms, and ROOT to put the histograms together. It also uses the DIM command "BeginRun" to coordinate with the ROOT FileSaver. The merger also works in background mode.

The ROOT FileSaver is also based on the ATLAS online packages, ROOT and DIM command. It uses the online software interface to fetch histograms, ROOT to dump histograms, and the DIM command to start the dumping action. The ROOT FileSaver is implemented as a plugin of the histogram server.

4 Testing and operation

For development and testing, a DQM emulator is designed to simulate a DQM server and the DIM command server. The DQM emulator can load both Monte Carlo and real data files, and send DIM commands. It is a powerful tool to diagnose problems during the development of the DQMF. Once taking data, the emulator will be replaced with the DQM server. Fig. 4 shows a diagram of the communication between the emulator and a client.

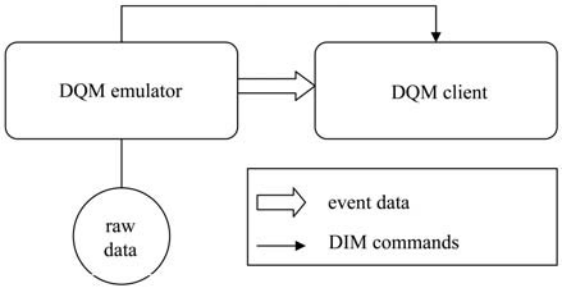

Fig. 4. Communication between the emulator and a DQM client.

The DQM emulator is developed in the online environment and implemented with a combination of DIM command and DQM server. The emulator uses the RawFileReader [10] interface to load raw files, the TCP/IP socket to transfer event data, and the DIM command to simulate the detector states. When reaching the end of one raw file, the state signals "EndRun", "BeginRun" and "Run" are sent successively at an interval of several seconds.

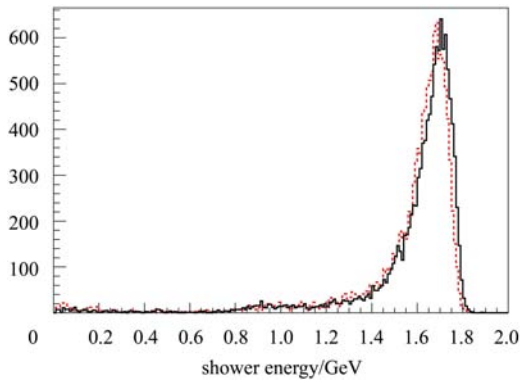

Fig. 5. The deposited energy of Bhabha events in the electromagnetic calorimeter.

Here, we present two example histograms produced by the DQMF in real operation. Fig. 5 shows the deposited energy of Bhabha events [11] in EMC, and Fig. 6 shows the energy loss of charged particles ($\mathrm{d}E/\mathrm{d}x$ [12]) in the MDC. The black (solid) histograms, compared with the red (dashed) reference histograms, show a normal EMC status (see Fig. 5) and an abnormal MDC status (see Fig. 6), respectively. The peak shift of $\mathrm{d}E/\mathrm{d}x$ is caused by improper calibration constants of MDC. Besides the above two illustrations, the DQM system also provides an abundance of histograms for different sub-detectors.

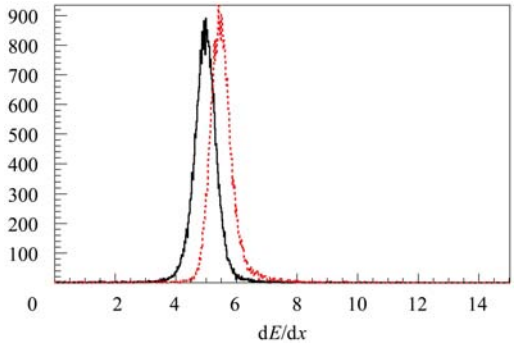

Fig. 6. The $\mathrm{d}E/\mathrm{d}x$ of charged particles passing through the main drift chamber.

5 Conclusion

The DQMF, which has been developed for the BESIII experiment, provides a visual, straight and quick solution for data quality assessment. DQMF adopts the C/S structure,sampling mode and full-reconstruction flow, etc, to achieve the experimental requirements. The first released version of the DQM framework has been available since September 2008, and after several updates of the DQM framework itself, more and more new physics algorithms were gradually added to monitor the detector. The released version has been running stably, and the results produced by the DQM system indeed reflect the changes in online and offline configuration in time, and offer a reliable reference to the shift crew.

The authors gratefully acknowledge Tiao Haolai for the contribution to the EventDisplay and Zou Jiaheng for the contribution to the RawFileReader.

References

1 Asner D M et al. (BESIII collaboration). International Journal of Modern Physics A, 2009, supp**24**(1): 24–25, 29
2 LI Fei et al. Online data processing and analyzing in BESIII DAQ in the 16th IEEE-NPSS Real Time Conference - Conference Record, 2009. 458–460
3 LI Wei-Dong, LIU Huai-Min et al. The Offline Software for the BESIII Experiment. Proceeding of CHEP06. 2006
4 ROOT, An Object Oriented Data Analysis Framework. http://root.cern.ch/drupal/
5 JIA Lu-Kui, MAO Ze-Pu et al. Chinese Phys. C (HEP & NP), 2010, **34**(12): 1866–1873
6 HU Ji-Feng et al. HEP & NP, 2007, **31**(10): 893–899 (in Chinese)
7 HE Miao. Simulation, Reconstruction and Low Momentum μ/π Identification of the BESIII EMC, in the XIV International Conference on Calorimetry in High Energy Physics, 2010
8 http://dim.web.cern.ch/dim/
9 Kolos S et al. A Software Framework for Data Quality Monitoring in ATLAS. International Conference on Computing in High Energy and Nuclear Physics (CHEP' 07), 2008
10 ZOU Jia-Heng. Dissertation in Shan Dong University, 2009. 43 (in Chinese)
11 LIU Chun-Xiu. Energy Calibration of the EMC with Bhabha Events at BESIII in the XIV Interenatial on Calorimetry in High Energy Physics, 2010
12 CAO Xue-Xiang, LI Wei-Dong et al. Chinese Phys. C (HEP & NP), 2010, **34**(12): 1852–1859