Online Data Monitoring in the LHCb Experiment

O.Callot\textsuperscript{1}, S.Cherukuwada\textsuperscript{2}, M.Frank\textsuperscript{3}, C.Gaspar\textsuperscript{3}, G.Graziani\textsuperscript{4}, E.van Herwijnen\textsuperscript{3}, B.Jost\textsuperscript{3}, N.Neufeld\textsuperscript{3}, M.Pepe-Altarelli\textsuperscript{3}, P.Somogyi\textsuperscript{3}, R.Stoica\textsuperscript{5}

CERN, Geneva, Switzerland

Markus.Frank@cern.ch

Abstract. The High Level Trigger and Data Acquisition system selects about 2 kHz of events out of the 40 MHz of beam crossings. The selected events are sent to permanent storage for subsequent analysis. In order to ensure the quality of the collected data, identify possible malfunctions of the detector and perform calibration and alignment checks, a small fraction of the accepted events is sent to a monitoring farm, which consists of a few tens of general purpose processors. This contribution introduces the architecture of the data stream splitting mechanism from the storage system to the monitoring farm, where the raw data are analyzed by dedicated tasks. It describes the collaborating software components that are all based on the Gaudi event processing framework.

1. Introduction
LHCb is a dedicated B-physics experiment being prepared at the LHC collider at CERN \cite{1}. LHC is scheduled to begin operation in 2008 and will deliver proton-proton collisions at a centre of mass energy of up to 14 TeV to the LHCb detector, at a rate of 40 MHz. The response to particle collisions from about 1 million channels will be sent at an expected rate of 1 MHz from roughly 300 front-end boards through 700 readout links into a software based High Level Trigger system (HLT), which reduces the event rate to roughly 2 kHz stored for later data analysis. To ensure data integrity and to detect malfunctioning components early, both the hardware and the software components involved in the data taking process are monitored and the collected data are analyzed. The output of the monitoring activity comprises summary information and statistical data in the form of histograms or scalars, such as counters. Anomalies such as unexpected distributions indicate malfunctioning components. In the following sections the mechanisms and techniques to perform the online verification of the collected data will be discussed.

---

\textsuperscript{1} LAL, Orsay, France
\textsuperscript{2} This research project has been supported by a Marie Curie Early Stage Research Training Fellowship of the European Community’s Sixth Framework Programme under contract numbers MEST-CT-2004-007307-MITELCO and \textsuperscript{3} MEST-CT-2005-020216-ELACCO
\textsuperscript{3} CERN, Geneva, Switzerland
\textsuperscript{4} INFN, Firenze, Italy

\copyright 2008 IOP Publishing Ltd
2. Hardware Configuration and Software Environment
As shown in figure 1, the front end readout boards (TELL1 boards) send data from particle collisions with a rate of roughly 35 GB/s through a switching network to the HLT farm nodes, where dedicated algorithms compute the decision on whether the event is to be accepted. Events with a positive decision are sent to the storage system and subsequently to the GRID for later offline analysis. In the storage system the data stream is partially duplicated and a fraction of the events are directed to a dedicated farm of some tens of commodity processors. These processors, called the monitoring farm execute tasks which monitor the detector response. The LHCb data acquisition system allows the independent configuration of up to 16 instances called partitions which may coexist. One or several subdetectors, a configurable number of subfarms and monitoring nodes are assigned to a single partition. The monitoring farm is much smaller than the HLT farm and shared between all concurrent partitions. Hence, an appropriate mechanism to support sharing is necessary.

![Diagram of the hardware layout. The HLT farm consists of up to 2000 processing elements. The monitoring farm on the right is comparably small with an expected size of ~20 processing elements.](image)

Figure 1: The hardware layout. The HLT farm consists of up to 2000 processing elements. The monitoring farm on the right is comparably small with an expected size of ~20 processing elements.

All front-end boards are equipped with a compact form-factor 486 compatible processor, which is interfaced to the hardware and allows limited data monitoring by reading certain registers. The components of the readout network can be monitored using a command line interface and/or SNMP. The following sections focus on high level data monitoring in the HLT and the monitoring farm, which consists of:
- Raw data monitoring to detect problems such as malfunctioning of individual detector components resulting in dead or noisy channels.
- Data monitoring using high level physics objects obtained from partial or full reconstruction of the raw data which allows verifying the overall performance and functioning of the apparatus.

Monitoring information in the HLT farm comprises mostly information obtained from rejected events-summary data which cannot be reproduced in the offline environment after the data taking activity is completed. In the monitoring farm summary data are produced to provide information quickly during data taking allowing the analysis and resolution of possible problems. In both cases, the summary data are distributed over many processes and need to be the collected and merged appropriately.

The goal is to design an open architecture in which the basic building blocks of the data acquisition are kept as independent as possible. These building blocks include implementations of functions to
abstract differences between operating systems (Linux, Windows), the buffer manager implementation to access the data, and the network data transfer mechanism. This approach results in flexibility when tailoring the necessary applications. All applications comply with the Gaudi component architecture [2] used also to analyze data offline.

Using either existing Gaudi components or replacing offline components with implementations specialized for the online environment, this choice facilitates:

- the customization and parameterization of the processes,
- the transparent access to event data in a manner identical to the offline environment,
- the redirection of monitoring entities such as histograms and output logging and,
- the embedding of data monitoring algorithms, which are developed in the offline environment.

3. Monitoring Results and Summary Data

Statistical monitoring information typically is produced by many instances of the same application. In the HLT this is one type of application, the event filter processes. In the monitoring farm however, a variety of different processes are creating their own set of information. Hence, initially the summary data are distributed over all these processes. As such, the monitoring information is neither appropriate for automatic analysis nor for visual analysis by the shift crew. To be presented, the information has to be collected and summed. The collection and summing of the statistical data is discussed in this section.

![Diagram of monitoring information flow]

**Figure 2:** The flow of monitoring information produced by various processes is summed by “Adders”. The Adders publish the summed information and make it available to clients responsible for display or persistence.
The monitoring applications use a Gaudi based online monitoring component [3] to publish monitored entities like histograms, scalars, etc. All tasks producing summary information publish these entities using the DIM protocol [4][5]. Each item published by a process is identified by a unique name within a process preceded by a unique process identifier [6]. Dedicated tasks called Adders sum the corresponding items published by a set of monitoring tasks and in turn publish the summed information themselves. Monitoring information is pushed to the Adder tasks at regular intervals to obtain coherent snapshots of the published information. Such snapshots are created by the monitoring applications at regular intervals during the data taking activity and at a run change. The monitoring applications detect a run change from the analyzed data and automatically reset counters and histograms. Hence, no external stimuli or commands are necessary to reset histograms. The Adder tasks sum the information from identical instances of tasks according to an agreed naming convention in the DIM namespace to ensure that only related counters, histograms etc. are combined. The Adders require a mapping between the various monitoring task types and the task instances, further knowledge of existing items per type to be combined from these tasks is explored at run-time. To sum the monitoring information from many sources, as in the HLT farm, a tree like structure of Adder tasks may be constructed. This tree approach is highly scalable at a comparably small price of adding latency to the summed data. Saver tasks subscribe to the information published by the top level Adder and save the monitoring information to disk. The Saver tasks update the information on disk regularly with a final version written at the end of a run. At the end of the run the Histogram Analyzer tasks are activated to search for anomalies in the collected distributions and to compare the summed information with reference data. In the event of irregularities an alarm is issued and displayed by the experiment control system [7]. Technically histogram analysis can as well be implemented as a continuous activity during data taking by subscribing to the summed information published by the Adders. An interactive program called the Presenter allows viewing of the summed monitoring information. It can work either in an online mode by subscribing to data published by running processes like, for example, the Adders, or in history mode using saved files. Besides being an ad-hoc browser with fitting functions and simple reference comparison, the Presenter may store and retrieve predefined monitoring views from the Histogram Database.

4. Partitioning of the Monitoring Farm
The processor nodes of the monitoring farm are shared between concurrent DAQ partitions. A simple solution would be to allocate one or several nodes as non shareable resources to a single partition – with a very rough granularity. To avoid such limitations and to maximize data taking efficiency, data monitoring with limited resources must fulfill the additional requirements:

- Data monitoring in a dedicated farm must not interfere with the data taking activity; monitoring is parasitic and should place minor constraints on the data taking performance.
- CPU resources should ideally be equally distributed between the monitoring tasks.
- Monitoring tasks of a given type may execute in several instances if the analysis time per event is significant and reasonably high statistics is required.
- Monitoring processes are known to perform differently on events with different physics content. Whereas noise detector spots are likely to be spotted more easily with random triggers, more sophisticated monitoring, which uses reconstructed tracks and vertices will perform better if specific physics channels can be preselected. Such channels or stream types can be specified by asking for events which satisfy a given trigger mask set by the HLT event filter processes.

To realize partitioning based on the requirements listed above, the nodes available for data monitoring are divided into a predefined number of logical slots (see figure 3), where each slot represents a non sharable resource to be allocated by exactly one partition and be populated by exactly one monitoring task. The required tasks are then distributed over the allocated slots.
After the distribution of the monitoring tasks, the input stream types required by each monitoring node can be computed. To avoid routing all input streams from the storage system to all nodes, groups of tasks which require the same type of events are distributed among adjacent slots, which are likely to be located on the same node.

The access to event data was realized using a producer-consumer pattern [8] as shown in figure 4a. The producer task is responsible for reading data from a network connection. As soon as the read operation is finished, the data block is declared to a managed shared memory area of the processor, the Buffer Manager (BM), and the producer is ready for the next read operation. The consumer task is activated each time an event is declared in the BM. The consumer is responsible for releasing the space occupied as soon as it has either sent the data to another producer or finished processing the data. Thus, the two activities of reading and processing can proceed asynchronously, provided there is always sufficient space in the buffer to accommodate at least one read operation. A BM is realized as a dedicated task executing on each node, which creates and initializes the shared memory. To avoid interference between different partitions, each partition manages its own set of buffer managers depending on the allocated monitoring slots.

The transport of event data to the nodes of the monitoring farm reuses this producer-consumer pattern (figure 4b). Sending and receiving data is separated into independent tasks running asynchronously in order to derandomize the flow of events. The task sending data is a BM consumer, the receiving task a BM producer.

The input streams to the monitoring nodes are fed by the storage system, which receives all events accepted by the HLT. However, a direct data connection to each monitoring node would put a large overhead on the storage system. To minimize the interference with the storage system, events are first sent to one or several intermediate relay nodes which act like a gateway as shown in figure 5.
The sender tasks on the storage system transfer the data to receiver tasks in the relay nodes. One sender-receiver pair is necessary per stream type and relay node. The trigger mask computed by the HLT processes can be used to selectively request events from the buffer manager without the need to read the data. Data senders executing on the relay node(s) then distribute the events to the monitoring nodes, where the receivers declare the data to a BM. Whenever new data are present, the monitoring processes are notified. The monitoring processes may then access the event data like any other BM consumer.

![Flow Diagram](image)

**Figure 5**: The flow of events from the distribution clients located on the storage system to the monitoring nodes.

5. **Task Control**

All tasks producing, summing or saving monitoring data participate in the data taking activity which normally proceeds in cycles. First all hardware components and tasks participating are configured. This is followed by a data taking period and finally a closedown phase. In the LHCb data acquisition system, all participating components are represented by a finite state machine to ensure that each is set to its correct state at each step of the cycle.

At the highest level the operator issues commands to the Run Control, which is responsible for the overall orchestration of the state transitions of the controlled subcomponents. The Run Control shown in figure 6 solely deals with macroscopic entities like units representing subdetector hardware or a control unit called *DataFlow*, which represents all Gaudi based processes participating in data taking. Internally the *DataFlow* unit consists of the sub-units *Monitoring*, *Storage* [9] and the HLT farm. The state diagram and the various transitions of the participating processes are described elsewhere [8].
Figure 6: The view of the control hierarchy as seen by the operator.

Tasks producing monitoring information and the corresponding Adders are grouped to form subsystems. A subsystem is associated to exactly one partition, which is controlled by one instance of the Run Control. Such subsystems are an HLT subfarm, the ensemble of subfarms allocated by one partition or the ensemble of slots in the monitoring farm allocated by one partition. The Saver for each partition task is associated to top-level subsystems such as the HLT farm or the monitoring farm.

6. Conclusions
Data monitoring is essential to detect malfunctioning detector components of the experiment. High level monitoring information is produced in LHCb either in the HLT farm or in the monitoring farm and is highly distributed over many processing elements. A scalable approach has been developed which allows the collection, analysis and storage of this information. Monitoring of the data of events accepted by the HLT is performed in a dedicated shared processing farm. A flexible sharing mechanism has been developed to allow several DAQ partitions to use this farm simultaneously.

References
[1] LHCb Collaboration, “LHCb Technical Proposal”, CERN-LHCC-98-04, February 1998.
[2] G. Barrand et al., “GAUDI - A software architecture and framework for building LHCb data processing applications”, CHEP 2000, Proceedings, Padova, Italy, February 2000.
[3] P. Vannerem et al., “Distributed Control and Monitoring of High Level Trigger Processes on the LHCb Online Farm”, ICALEPCS’2003, Gyeongju, Korea, October 2003.
[4] E. van Herwijnen, “Control and Monitoring of on-line trigger algorithms using Gaucho”, CHEP 2006, Mumbai, India, 2006
[5] C. Gaspar et al., “DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication”, CHEP 2000, Padova, Italy
[6] D. Galli et al., “The Task Manager for the LHCb On-Line Farm”, LHCb 2004-099 DAQ
[7] C. Gaspar et al., “An Integrated Experiment Control System, Architecture and Benefits: the LHCb Approach”, IEEE Trans. On Nucl. Science, Vol. 51 (2004), Issue 3, Part 1, p. 513-520
[8] M.Frank et al., “The LHCb High Level Trigger Infrastructure”, CHEP 2007, Victoria, BC, these proceedings.
[9] N.Neufeld et al., “Data Stream handling in the LHCb experiment”, CHEP 2007, Victoria, BC, these proceedings.