# Physics and Data Quality Monitoring at CMS

C. Leonidopoulos*, E. Meschi, I. Segoni, *CERN, Geneva, Switzerland*
G. Eulisse, *Northeastern University, Boston, USA*
D. Tsirigkas, *Universität Zürich, Zurich, Switzerland*

*Abstract*

The Physics and Data Quality Monitoring framework (DQM) aims at providing a homogeneous monitoring environment across various applications related to data taking at the CMS experiment. Initially developed as a monitoring application for the 1000 dual-CPU box (High-Level) Trigger Farm, it quickly expanded its scope to accommodate different groups across the experiment. The DQM organizes the information received by a number of monitoring producers and redirects it to monitoring consuming clients according to their subscription requests, in the classic publish-subscribe paradigm. Special care has been given to the modularity and stability of the system with the clear separation of the production of the monitoring information from the distribution and processing. We describe the features of the DQM system and report on first measurements of its performance on a small subfarm prototype.

## INTRODUCTION

The DQM infrastructure has been designed to be flexible and easily customizable so as to be usable by different groups across the CMS experiment at LHC [1]. Applications that can benefit from a unified approach to monitoring range from the high-level trigger algorithms in the Filter Farm [1] to data integrity and hardware performance checks by sub-detector groups up to off-line reconstruction jobs carrying out "production validation". The primary goal of the DQM system is however to guarantee the quality of physics data collected by the general data acquisition.

A generic interface (independent of the specific technology implementation) is provided for the creation and update of *monitoring elements* (*e.g.* histograms or simple numbers), allowing direct insertion of monitoring statements in the reconstruction code. Monitoring producers publish a list of available information (*monitorables*) to be delivered to monitoring consumers upon connection. These producers accept subscription requests for delivery of monitoring information in regular updates. The DQM infrastructure provides functionality to collect and organize information received from a number of producers, and redirect it to consumers according to their subscription requests. This interface can be accessed from standalone programs, or can be used from within reconstruction applications and modules. On the client side, tools are provided for evaluating the consistency of received information to reference information retrieved from a database. Clients can also update these references, set thresholds, raise alarms, and create error messages for use by the central error logging facility.

## ARCHITECTURE

The DQM framework is designed to deal with sets of monitor elements from the creation in monitoring producers (*sources*), to the organization and redistribution, on a periodical basis, in the *collectors*, to their final use by monitoring information consumers (*clients*). Sources are defined as individual nodes that have either direct access to or can process and produce information we are interested in. The creation and update of monitor elements at the source can be the result of processing input event data (event consumers) or input monitor elements (monitor consumers).

At the other end of this architecture are the clients. Clients get automatically notified of monitorables from all sources combined. They can subscribe to and receive periodic updates of any desired subset of the monitorables, in a classic implementation of the "publish-subscribe" service. A hierarchical system of collector nodes is responsible for the communication between sources and clients (*e.g.* subscription requests) and the actual monitoring transfer. These nodes serve as collectors for the sources and as monitoring servers for the clients.

In order to minimize the interference with the "main" application running in the source process (*e.g.* analysis, calibration-alignment, trigger algorithm, etc), DQM operations at the source are reduced to a minimum. All CPU-intensive tasks (*e.g.* comparison to reference monitoring element, display, database storage, etc.) are to be carried out at the client's side.

The above design aims at

- shielding the sources from connecting clients that could slow down the main application or threaten the stability of the source
- facilitating the quick transfer of the monitoring information from the sources to the collectors.

To this end, sources are connected to only 1 collector each (but a collector can connect to multiple sources). Clients do not have direct access to the sources. All source-client communication is carried out through the collector (or collectors).

In this design, the production of the monitoring information is clearly separated from the collection and the

*Contact person: `Christos.Leonidopoulos@cern.ch`

[1]The Filter Farm is a 1000-dual CPU PC farm running the High Level Trigger algorithms at CMS. It replaces the "L2" and "L3" trigger elements of a traditional HEP experiment.

processing. The collectors act as the "middle man": they are responsible for advertising the monitorables to different clients and serve monitoring requests. The nature of the collection and processing of the monitoring information is statistical by construction. In particular, the DQM

- is meant to help the experts identify problems that occur (and are monitored) *over a period of time* and is not expected to be capable of spotting punctual problems
- does *not* give access to particular events
- does *not* guarantee that 2 clients will receive identical monitoring information

## COMPONENTS AND DATA FLOW

The DQM infrastructure supports various monitorable types. 1D-, 2D- and 3D-histograms, 1D- and 2D-profiles, scalars (integer and real numbers) and string messages can be booked and filled or updated anywhere in the context of reconstruction and analysis code. The infrastructure takes care of publishing, tracking updates, and transporting these updates to subscriber processes. The DQM infrastructure does *not* provide support for publishing or transport of individual event data. Distribution of data to "event consumers" is provided by a separate system at CMS, not discussed here [2]. Access to booking, filling and modifying monitor elements is provided via abstract interfaces in every component. monitor elements are organized in (UNIX-like) directory structures with virtually unlimited depth, from which monitor consumers can "pick and choose". In every component, it is possible at any point in time to create ROOT-tuples [3] with "snapshots" of the monitoring structure for debugging and reference.

### *Sources*

DQM services available to the source not only keep track of updates to existing monitor elements, they also enable dynamic modifications to the monitoring structure. The list of available monitoring information can be modified at runtime by booking or deleting monitor elements via the public DQM service interface. Updated information from an individual source is distributed to all consumers (through the collector via TCP/IP) by an update task (*MonitorDaemon*) running periodically in a separate thread of the source process. The interval between 2 MonitorDaemon updates, which defines a monitoring cycle, can be configured for each individual source process. A "reset" switch can be used to specify for each monitor element whether monitoring contents should be reset at the end of a cycle[2].

The MonitorDaemon maintains the connection with the collector and uses the DQM service to collect updates to be transmitted to it. The main application (which could be a critical one, like a HLT process), is not affected by the failure of a downstream component in the DQM system. As an example, the source can continue to run even if the connection to the collector is lost (*e.g.* the collector has crashed).

[2]This option should be turned on for monitor elements that describe dynamic content (e.g. hit occupancy of a sub-detector) and off for monitor elements that describe accumulating quantities (e.g. number of events processed, number of errors, or counters of rare events).

### *Collectors*

Collectors serve as dispatch points between sources and clients. Unlike sources and clients, collectors are completely standardized and do not need any customization. A collector accepts network connections from sources and clients (via TCP/IP). A source can post messages to the collector advertising available monitor contents. All connected clients are dispatched with the entire published content available at the collector. The collector receives subscription messages from clients that are relayed to the appropriate sources. When a source sends an update message containing new data, this is relayed to all subscribed clients. Individual sources and clients can be added or removed at run-time. The collector is responsible of keeping track of active connections.

### *Clients*

A generic DQM client application is distinct from a source in that it normally only deals with monitor data, and not with event data. Client input comes in the form of updates of subscribed information from one or more collector instances the client is connected to. As mentioned above, connections can be dropped without affecting the overall functionality of the DQM system and its sources. Standard components are provided that allow the client to:

- start and configure itself, making connections to the relevant collector(s)
- load an initial subscription list at configuration; this list can be later edited and saved
- be notified of the data taking configuration (DAQ configuration, trigger tables) and of run start and stop
- subscribe to selected subsets of data available from the connected sources
- add or remove available items from the subscription list at runtime
- receive and keep track of periodic updates of monitoring information from multiple sources
- collate information from different sources
- maintain a local list of available data; this includes all input data, results of collation, and all other monitor data created in the client itself
- create groups of monitor elements for analysis or display
- create and attach quality tests (*rules*) to be run on monitor elements: these rules are evaluated for each update and used for diagnostic reports

Client applications must be customized for the use by an individual subsystem.

### *Client customization*

As discussed earlier, the majority of the operations involving monitor elements takes place on the client side. Here we list a set of tools used to customize these tasks, accessible only by clients.

- Analysis tools for monitoring element operations: "reset", "accumulate", "collate", "compare to reference".
- Status flags to summarize with a single discrete parameter the status of hierarchical components of a subsystem. This is convenient for summary pages on a GUI that can give the overall status of components sub-detector *e.g.* through a color-coded system. Problem flags can be set according to rules, alarms raised or masked.
- Archival facility to store (and retrieve) custom sets of monitor elements to be "played back", used as reference, or for historical analysis.
- Graphical User Interface to give interactive access to the custom operations discussed in this section via a standard set of graphical interface widgets, and to provide visual feedback to the user (overall and hierarchical status display, representative plots).
- Display of arbitrary sets of monitor elements.

Generic graphic clients are also provided that can be connected to a given application, allowing the standard DQM interface to be visualized as complex live displays of monitor information (e.g. via a web interface).

## TIME MEASUREMENTS

We present here some preliminary time measurements taken from a mini-farm prototype set up at the CMS site. The tests consist of time measurements observed when shipping gaussian distributions stored in 50-bin float 1-D histograms over a 1 Gb ethernet connection using TCP/IP and `ROOT v5.08.00b` [3]. Fig. 1 shows the distribution for the time required to ship 1000 histograms from a single source to a collector. Fig. 2 shows the distribution of the time that is wasted until the connection with the collector has been established and the shipping can begin. Feeding this information back into the DQM system would be the first step towards reducing the waiting time before the updated monitoring information can be transferred, *e.g.* by adjusting the update rate. Table 1 is listing the shipping times for various numbers of histograms.

Table 1: Time required to ship histograms over a 1 Gb ethernet connection. See text for details.

| # of histograms | Shipping time (ms) |
|---|---|
| 10 | $< 0.1$ |
| 50 | $0.1 \pm 0.0$ |
| 100 | $0.2 \pm 0.0$ |
| 500 | $4.5 \pm 0.6$ |
| 1000 | $6.0 \pm 0.2$ |

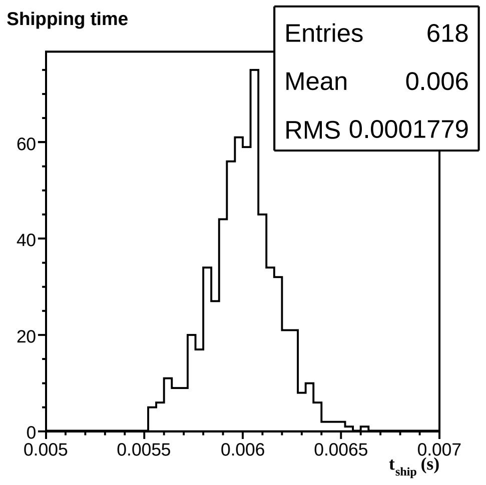

Figure 1: Distribution of shipping times for sets of 1000 histograms over a 1 Gb ethernet connection. The average time is $6.0 \pm 0.2$ ms.

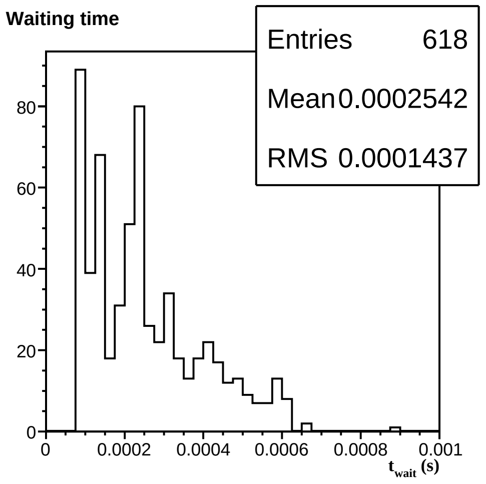

Figure 2: Distribution of waiting times for a simple system with one source, one collector and one client. The size of the tail of the distribution depends on the volume of data, the update rate and the number of nodes in the system.

## SUB-DETECTOR MONITORING

Implementation of DQM tools is under active development by the different sub-detector groups. The current focus has been on addressing the short-term operational needs of the Cosmic Challenge. For example, a Drift Tubes DQM histogram browser with a GUI based on IGUANA technology [2] has been used to monitor the drift tubes as part of the commissioning effort at the CMS site (SX5), shown in Fig. 3. This development will continue on to the development of monitoring programs for the long-term needs of the sub-detectors.

## REFERENCES

[1] `http://cmsinfo.cern.ch/Welcome.html`

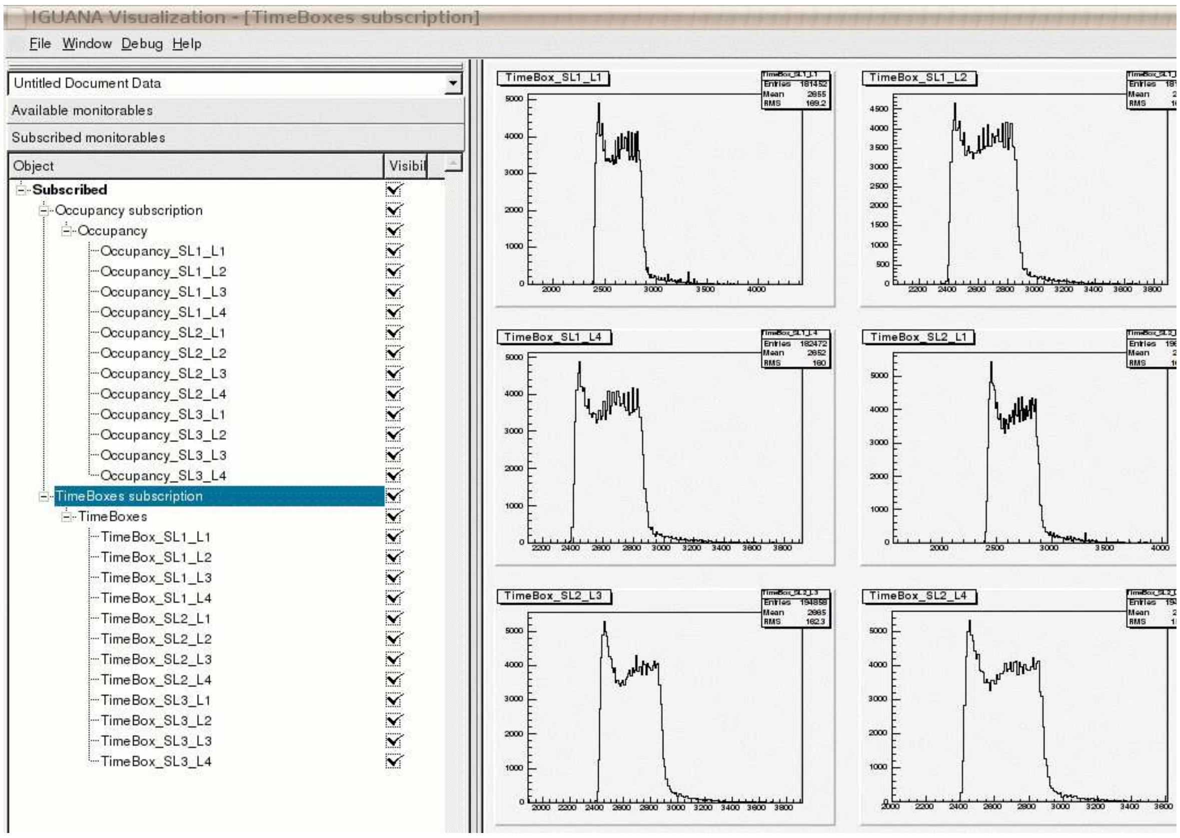

Figure 3: Screen shot of the Drift Tubes (DT) DQM browser of online data sources: occupancy and time boxes subscription, using the interactive IGUANA GUI and tree controller to display several embedded ROOT canvas components. The data sources are the cosmics muons taken at the CMS site (SX5) during commissioning of the installed DT.

[2] CMS uses the IGUANA technology for event display. See `http://iguana.web.cern.ch/iguana`

[3] `http://root.cern.ch`