Data Quality Monitoring for High Energy Physics (DQM4HEP)
Software overview - UI - Tests

Eté Rémi, Antoine Pingault

Université Claude Bernard Lyon 1 - Institut de Physique Nucléaire de Lyon / Ghent University

11 septembre 2015

Summary

Global software overview

DQM4HEP : an online monitoring system for data quality

Key points

- Standalone run control for DQM
- Data distribution system over the network : collector (server), client
- Data processing adapted to DQM purpose : analysis, standalone.
- Histograms distributing system over the network : collector (server), client
- Visualization interface (Qt GUI)
- Job control interface
- **Data model independent !** (optional build of LCIO streamer interface)
- Simple ELog interface

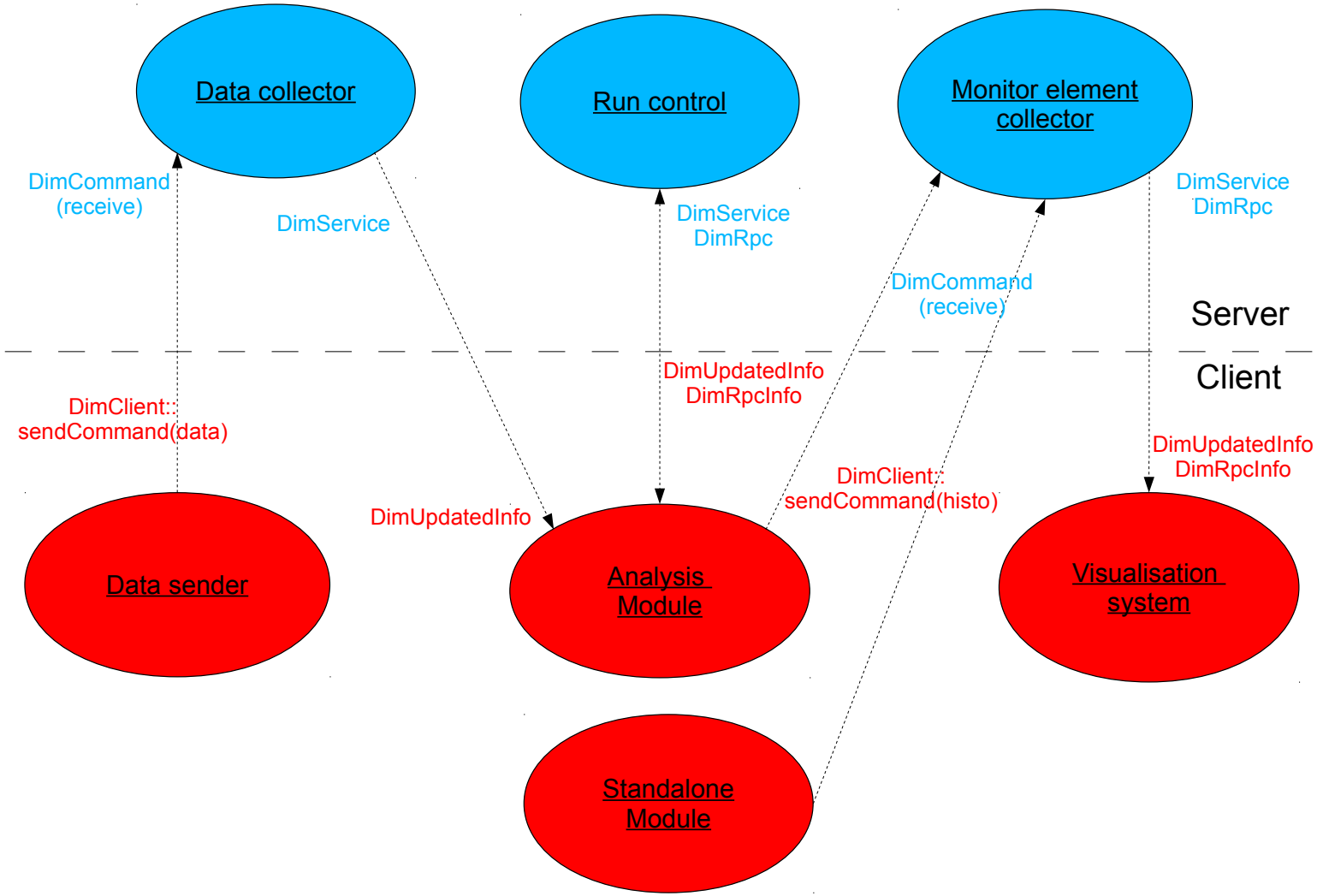
Dependencies

- ROOT
- DIM (+ DIMJC)
- Json cpp (config)
- Optional deps : LCIO, Qt, Elog bin

All in C++ ! !

Global software overview

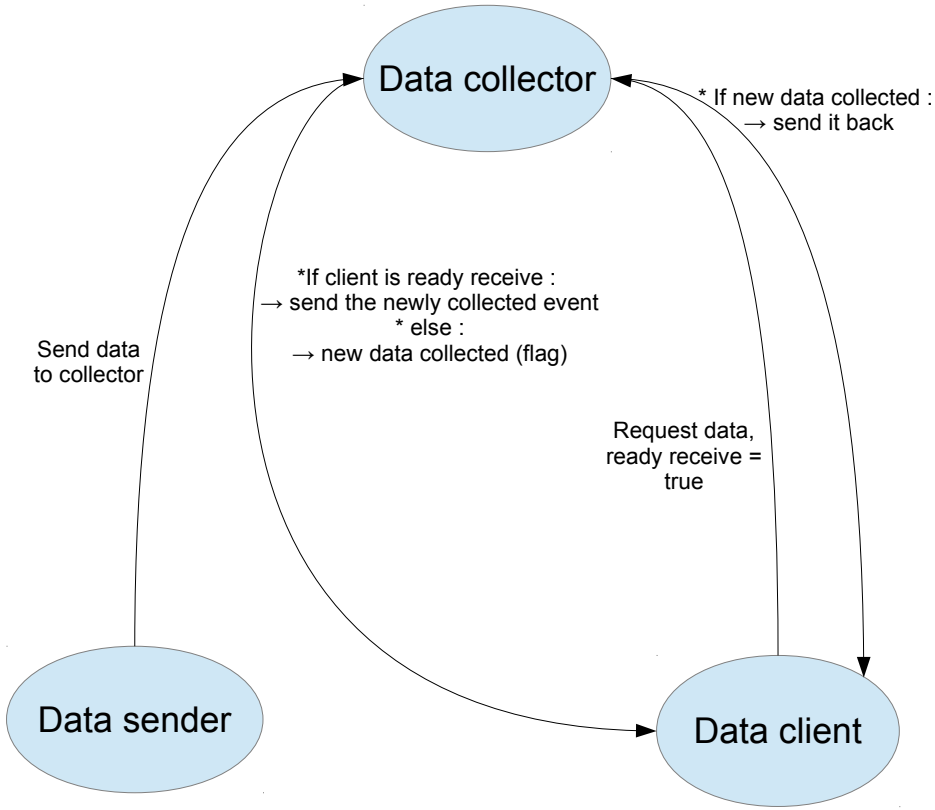
Network implementation



Global software overview

Data collector

- Collect data (storage)
- Publish data (service)
- Update and query client mode



Global software overview

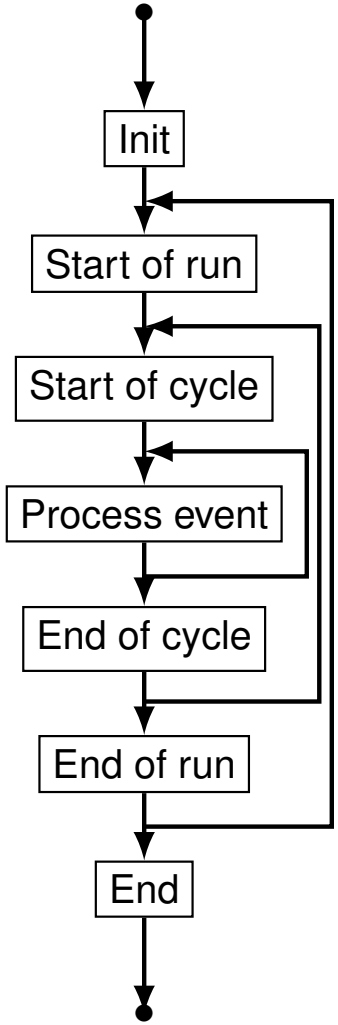
Module applications - analysis module

Purpose

- Query and process data
- Produce monitor elements (histograms, scalars, generic TObject)
- Run structure (SOR, EOR)

- **Init** : Initialize the application : load dlls, declare services, etc ... Wait for a SOR
- **Start of run** : start cycles loop, open archive
- **Start of cycle** : start a cycle of data processing
- **Process event** : Process incoming data, fill monitor elements, etc
- **End of cycle** : collect monitor elements and send, write archive.
- **End of run** : Wait for SOR, close archive.
- **End** : Clean and exit module.

Designed for data analysis (raw data, energy rec, tracking, PFA, etc ...)



Analysis module application flow

Global software overview

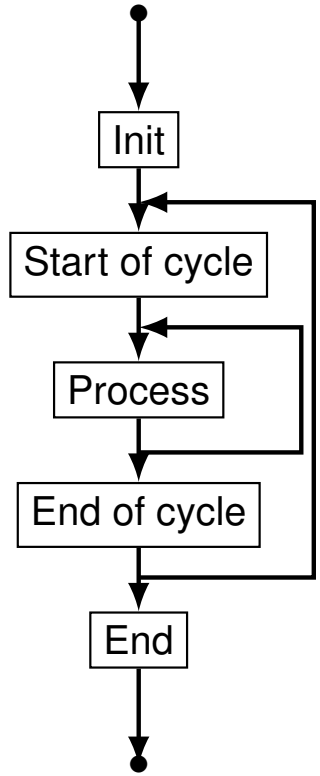
Module applications - standalone module

Purpose

- No data query
- No run structure
- Produce monitor elements (histograms, scalars, generic TObject)

- **Init** : Load dlls, init the module.
- **Start of cycle** : start a timer cycle of n seconds
- **Process** : call back function.
- **End of cycle** : collect monitor elements and send
- **End** : The application has received a signal to exit and the process ends.

Designed for *slow control - like* data processing.

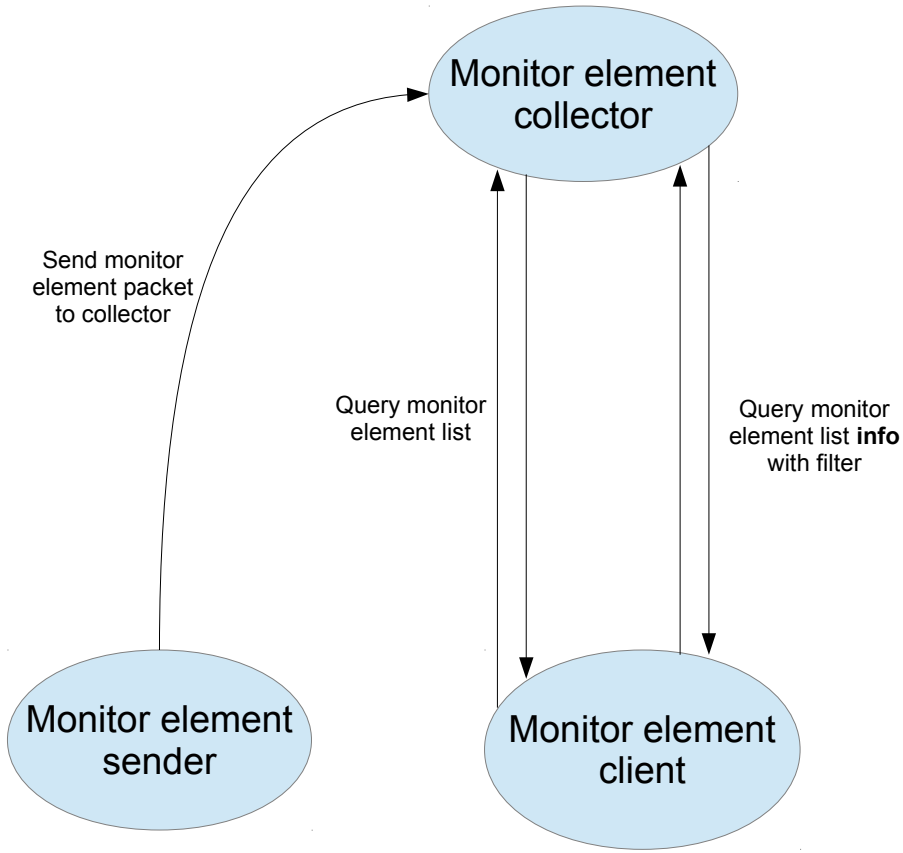


Standalone module application flow

Global software overview

Monitor element collector

- Collect monitor elements (histograms, scalars, generic TObjects)
- Redistribute them (service)
- Query mode only



User Interface

Streaming interface

The framework has no dependency on the type of data transferred over the network !

The streamer for LCIO is implemented and provided as a plug-in in the software. The type of data that is transferred over the network can be user defined.
Users have to implements the `DQMStreamerPlugin<DQMEvent>` interface :

```
virtual StatusCode serialize(const DQMEvent *const pObject, DQMDataStream *const pDataStream) const = 0;
virtual StatusCode deserialize(DQMEvent *&pObject, DQMDataStream *const pDataStream) const = 0;
virtual DQMPlugin *clone() const = 0;
```

The `DQMDataStream` class provides a simple API to read and write raw buffers. The streamer is retrieved by using the DQMPluginManager singleton.

```
DQMPlugin *pPlugin = NULL;
DQMPluginManager::instance()->getPlugin("MyStreamerName", pPlugin);
// or
DQMPluginManager::instance()->getPluginClone("MyStreamerName", pPlugin);
// or directly
MyStreamer *pStreamer = NULL;
DQMPluginManager::instance()->getCastedPluginClone<MyStreamer>("MyStreamerName", pStreamer);
```

For an experiment with simple data structures, it can be useful to define a raw data streamer and propagate the data through a monitoring system without taking care about the network interface.
The streamer has to be compiled in a shared library (.so or .dylib). The plugin is then loaded using the environment variable DQM4HEP_PLUGIN_DLL :

```
export DQM4HEP_PLUGIN_DLL=libMyStreamer.so # or .dylib
```

User Interface

Module API

- Implementation : subclass DQMAnalysisModule or DQMStandaloneModule
- Env var DQM4HEP_PLUGIN_DLL (Marlin-like)

- Modules operation : `DQMModuleApi`
  - Monitor element management (booking, removing, mapping access, etc ...)
  - Directory structure (mkdir, cd, rmdir, etc ...)
  - Quality test processing (add, run, report)

Directory structure, monitor elements and quality test reports $\rightarrow$ ME collector.

User Interface

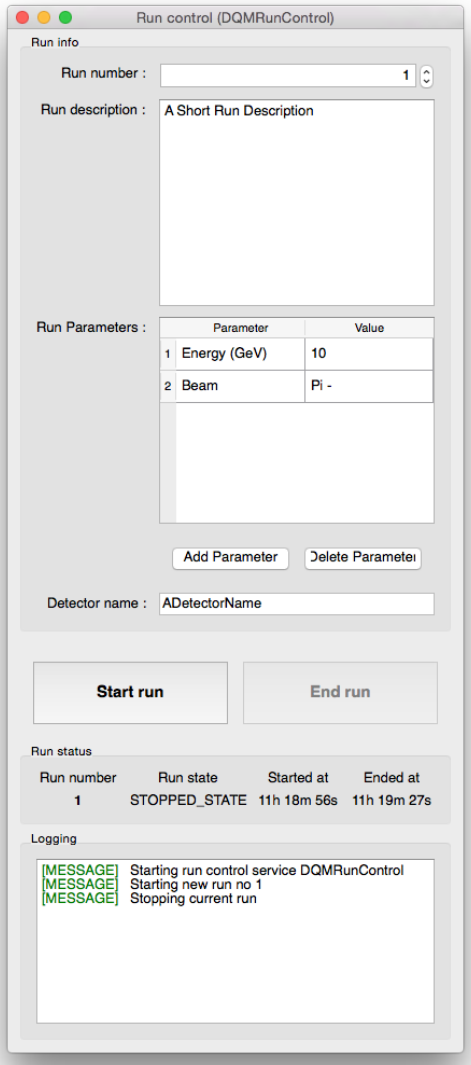
Gui visualisation

Gui interfaces for DQM client developed :

- Run control, job control, online monitoring
- Written with Qt4 framework 
- Easily configurable with json.

User Interface

Run Control GUI



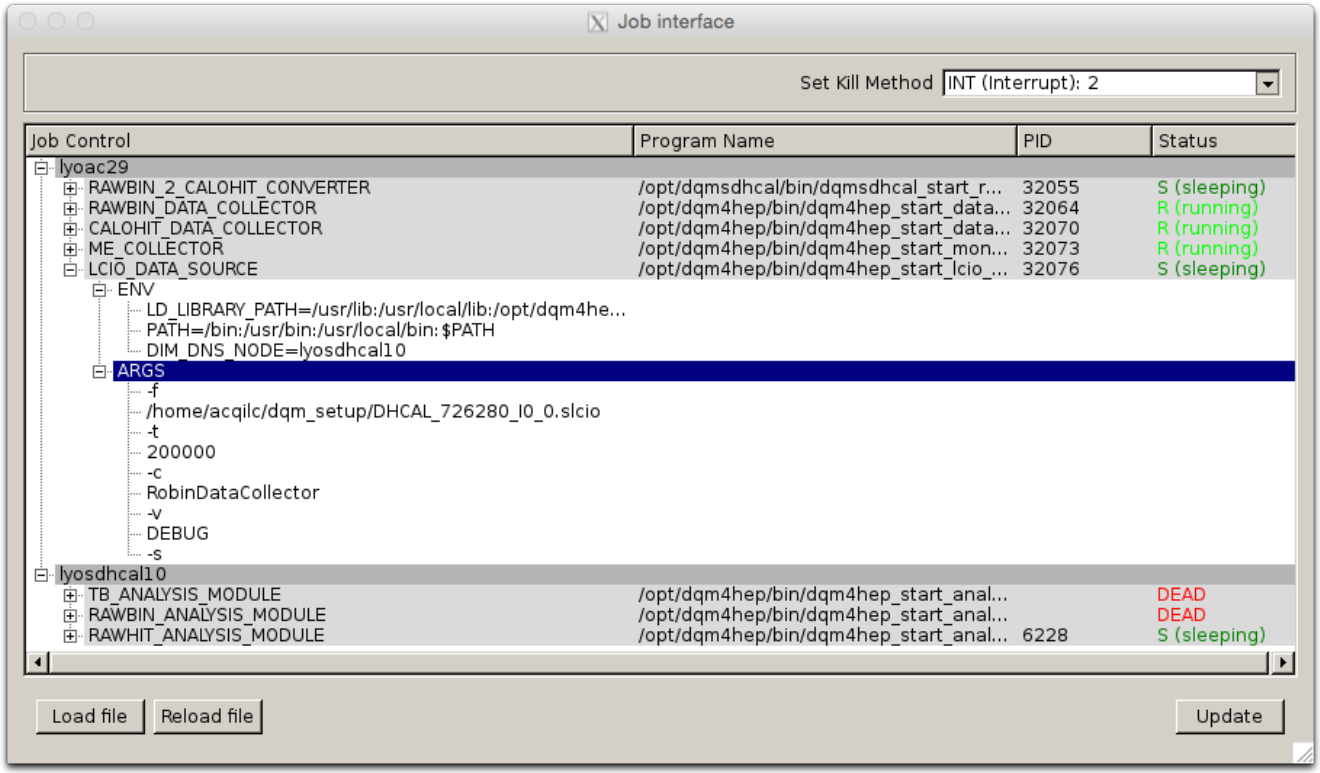
Run Control GUI

- Send SOR and EOR signals with run number, detector name, run description and parameters
- Control run status (State, Started/Stopped time)
- Every action is logged for easy information overview

User Interface

Job Control GUI

Job Control GUI

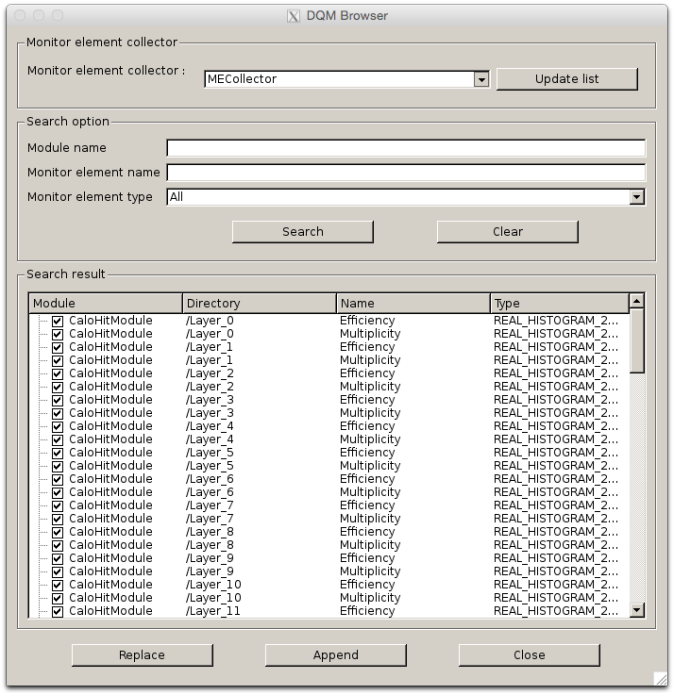


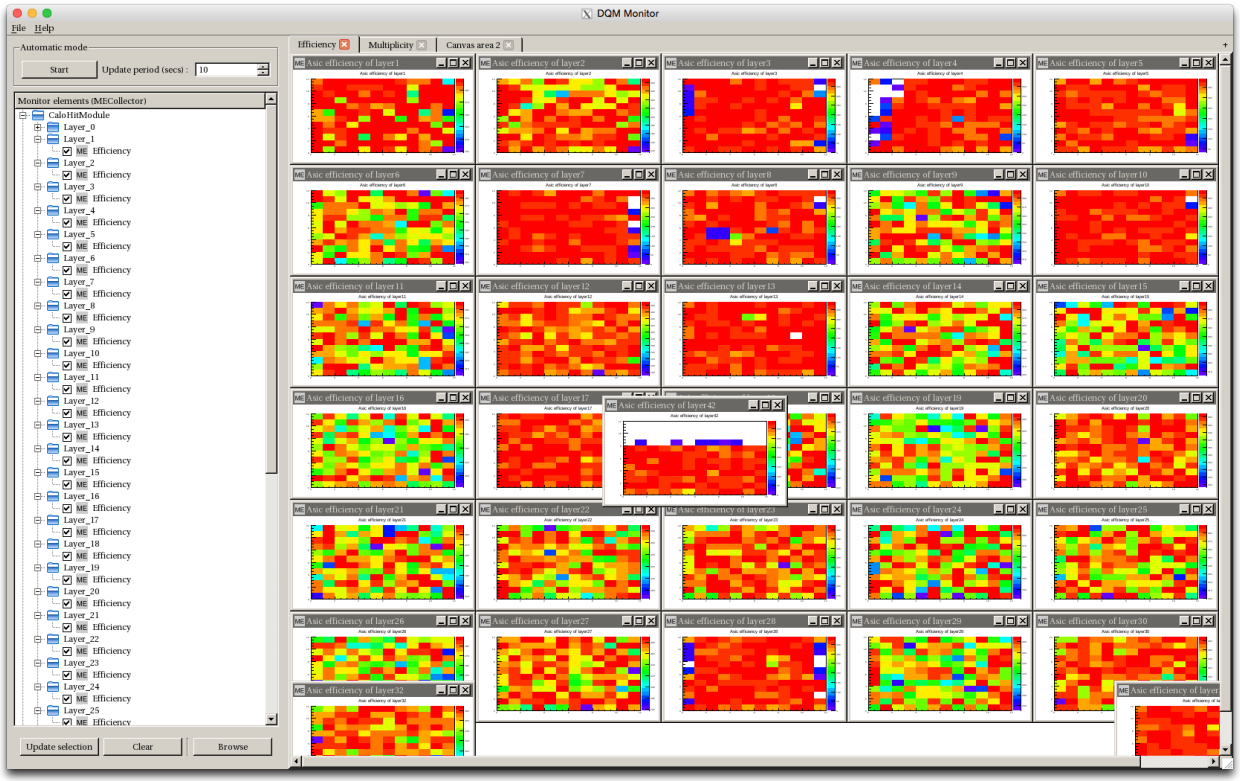
- Load a list of modules and their parameters (Host, name, data files etc.) and display their status on the network
- Configuration loaded from Json files
- Manage the modules (Start, Kill, etc.) on all hosts running a job control daemon (dim)

User Interface

Monitoring Gui

Monitoring GUI





- Query interface (Browser) to build a selection of data to monitor
- Multiple Canvas area available
- Not simple images but ROOT Object -> Interaction (Zooming/Fitting etc.) possible

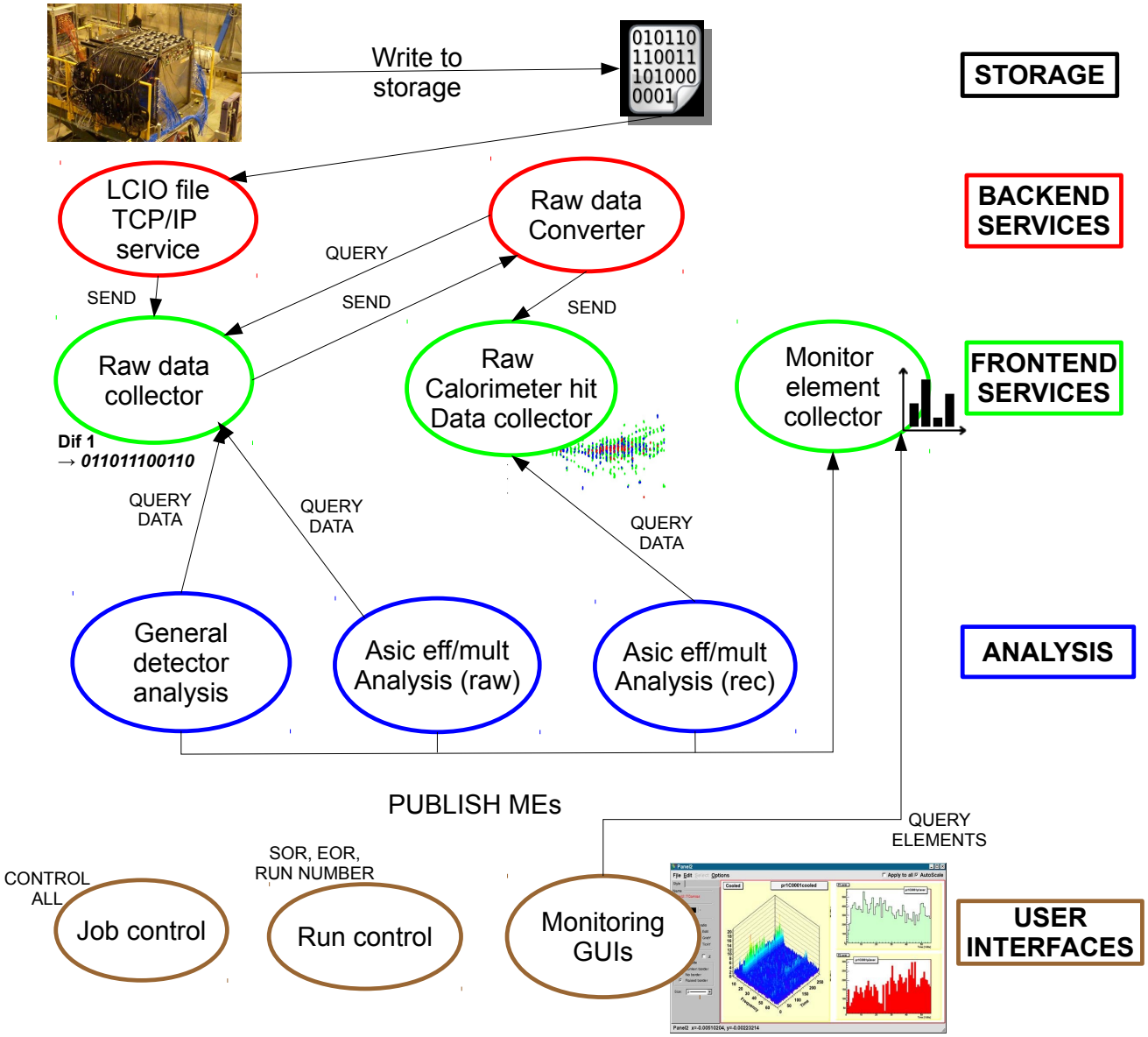
DQM4HEP tests

Tests of the framework were performed using the SDHCAL electronics (SDHCAL Difs).
Ideas were to test :

- Easy to deploy ? configure ?
- Easy to use interfaces ?
- Network saturation with many analysis
- Test specific SDHCAL DQM tools :
  - Raw data converter service (Streamout)
  - Event reconstruction tool (Trivent)
  - Online analysis
  - Online data taking (/dev/shm reader)

DQM4HEP tests

Deployment



DQM4HEP tests

Results

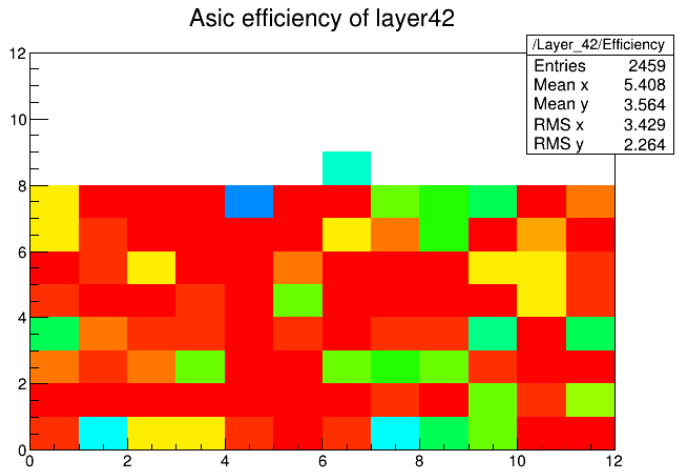
- Easy package installation/update (CMake + GIT)
- Input file : SDHCAL TB December 726280 (20 GeV pion run)
- Job control really makes life easier !

Easy to start, stop, restart and reconfigure programs. Easy to detect configuration problems !

Gui can display :

- total efficiency mapping (47 plots)
- total multiplicity mapping (47 plots)
- global efficiency/multiplicity and statistics (4 plots)

at the same time, with frequent updates (every 5 seconds) without lagging. Gui handling still ok.



Disconnected DIF in layer 42 !

More tests :

- Static code analyser (cppcheck) : no errors, POSIX norm ok
- Online data taking (/dev/shm) : OK !

DQM4HEP tests

Conclusions and plans

Conclusion

- Network decouples/links different part of the system.
- Plugins (modules, data streaming) to configure and run the system
- Tools for data feed in the system from the DAQ (event client interface)
- GUIs to control/monitor the system
- Tests OK, numbers ... ?

Plans

- Full implementation of SDHCAL DQM :
  - Gas system, HV, LV, T/P
  - Global detector (efficiency, multiplicity, nHit 1-2-3, rate)
  - Particle ID (counting, selection)
  - Particle specific modules (pion, electron, muons)
  - Energy reconstruction
  - Particle flow (performance)
- Performance tests : timing, statistics, compression (network, streaming)
- Reimplementation of some interfaces (networking)
- ILCSOFT ?
- Combined ECAL test-beam : combined DQM ?

Thanks for your attention