Table of Contents

LHCb RICH Online Monitoring

Contact and mailing list

Two mailing lists are available : **lhcb-rich-software [AT] cern [DOT] ch** and **lhcb-rich-dataquality [AT] cern [DOT] ch**

The Online Cluster

Getting an account on the online cluster

The procedure is described on the Online TWiki The accounts are personal and not related to the "normal" CERN accounts (e.g. on lxplus), you'll get a new login and a separate password for the online machines. The default (supported) shell is bash. A group area exists in `/group/`, so for the RICH group, our area is in `/group/rich`.

A log-in script is provided which sets the environment: `/group/rich/scripts/rich_login.sh`

Using the online cluster

The online network is separated from the normal CERN network, the only way to access the machines is via the gateways, please follow the instructions on the dedicated online page from the online group.

Once you are logged on, you can setup the LHCb software environment with

```
 > source /group/rich/sw/scripts/setup.sh
```

Which will give you access to the software installed on the online system. *Note that this command sets the `User_release_area` to the **GLOBAL RICH** area `/group/rich/sw/cmtuser`, used for online monitoring, etc. Changes in this area should be done carefully.*

For testing, you can use a different area by changing User_release_area by, e.g.

```
 > export User_release_area=$HOME/cmtuser
```

which will change it to use a cmtuser area in your private home area.

To setup an environment for a specific software package do:

```
SetupProject --build-env Application version
```

where

- `--build-env` sets up the necessary directory structure for further software development. This is not needed to just run a default released application
- `Application* is the name of the application to run, e.g. =Panoptes`
- `version` is the version of the application, e.g. `v1r5`

Access to the LHCb software repository (CVS) is described here. Note though **DON'T** use the SSH approach (as currently recommended on those pages) since this will lead to write access problems between different users. Instead run

```
 > kinit AFSUSERNAME@CERN.CH
```

where AFSUSERNAME is your LHCb AFS username, to get a kerberos token which will allow the standard kerberos access to CVS to work. You can check if you have a Kerberos token via `klist`

Once you have done this `getpack` will work as you are used to on lxplus etc. E.g.

```
 > getpack Hlt/Moore v2r2
```

To use CVS you will need to be on one of the gateway machines.

DIM DNS

The DIM system uses a DNS to locate available services. The service for the online systems is running on `mona08`. When testing algorithms or running offine in IP8, make sure that the environement variable `DIM_DNS_NODE` does **not** point to this computer.

CAMERA -Commissioning and Monitoring Error Reporting Application

Camera is a network application specifically built for communication and error logging for the commissioning and monitoring.

Using camera in the control room

Using camera in the control room

Algorithms reporting to Camera

Algorithms reporting to Camera

Camera Gui hints

Camera Gui hints

The Rich Raw Data Format and Decoding Software

The RICH raw data format is described in detail in the EDMS note accessible from this page which details all the LHCb data formats.

The decoding is performed by the Gaudi software implemented inside the Rich/RichDAQ CMT package. The primary method that performs he decoding is implemented in the Rich::DAQ::RawDataFormatTool Gaudi tool, specifically the method `decodeToSmartIDs (Rich::DAQ::L1Map &decodedData) const`.

The data is decoded into a data structure, `Rich::DAQ::L1Map` which is designed to map rather closely the structure of the L1 raw data itself.

Rich::DAQ::L1Map is a map containing the L1 board ID as the key, and the target another map `Rich::DAQ::IngressMap`. This Ingress map contains (up to) four entries, one for each ingress in a given L1 board, where the key is the ingrees ID and the target the a data object of type Rich::DAQ::IngressInfo.

The Rich::DAQ::IngressInfo data object contains two main data parts. The header word for that Ingress Rich::DAQ::L1IngressHeader and a data map which contains one entry for each HPD in that ingress, the Rich::DAQ::HPDMap. This entity is itself (yet) another map which has as its key the RichSmartID identifier for the HPD in question and as its target a Rich::DAQ::HPDInfo data object. The Rich::DAQ::HPDInfo is the final data object in the overall structure, and contains the decoded data for a single HPD, namely the header and footers when available and a vector of the RichSmartID identifiers for each active HPD pixel.

The above description makes the decoded data sound very complicated. In fact, it isn't really and the usage of this data is best done by simply looking at an example where the decoding tool is used. The follow section, which describes the RICH monitoring algorithms, provides various examples of this.

The Online Monitoring Software

The main application and package for the RICH online monitoring software is called Panoptes This package contains the executable for the online monitoring and offline tests as well as the main steering files

The following sections detail the (evolving) online monitoring software. **N.B** histograms and counters published to DIM should be declared in the `initialize()` method of the monitoring algorithm

Software tags / releases

The monitoring software is released from time to time with a given tag (or release number) to allow for easy comparison and interaction with other groups. The structure of the tag is as follows `vNrMpK` where `N` is the major release number, `M` the minor release number and `K` the number of the patch. The latter one is optional and may not be present.

Currently available tags are:

v1r0	Used with version v4r7 of the online software environment
v1r5	Used with version v4r12 of the online software, using in addition `Gaucho v6r2` for the new `MonObjects`

CMT Packages

The online-monitoring software for the RICH detector is split onto several packages:

`Rich/RichOnlineMonitors`	Gaudi-algorithms for the online monitoring, e.g. hitmap monitors, DAQ monitor, etc.
`Rich/RichMonitoringTools`	Tools for the online monitoring, e.g. CAMERA,
`Rich/RichMonitoringSys`	Steering package for the online monitoring, contains the options files, steering scripts etc. (but no source code)

Options files

The main options files are collected in the package `Rich/Panoptes`. This way, no source code needs to be checked out if only the options files need to be modified (once the Rich monitoring package are part of the online environment, at present, all packages need to be checked out from CVS). We're currently changing all configuration files to Python, the main steering file is `Panoptes-Default.py` in the `options` directory.

Currently implemented monitoring alogrithms

This algorithm monitors the number of hits in the active HPDs. Number of hits is published as a counter via PVSS. In addition, the same information is available as a 1D histogram per HPD (disabled by default). N.B. The algorithm is not yet operational pending discussions of when/where the relevant entities should be published to DIM via `declareInfo()`. The algorithm is controlled via the following options

MonitorRate	monitor every nth event
FillHistos	fill histograms in addition to publishing counters
ExtendedVerbose	even more output than `verbose()`
HistoNHitMin	lower left border of histogram showing number of hits
HistoNHitMax	upper right borer of histogram showing number of hits
HistoTrendStep	number of steps in trend histogram

Provides hit-map histograms of each RICH detector and each HPD panel

DoAliceMode	fill histograms in ALICE mode
OutputLevel	verbosity level

N.B. When running in alice mode, the lhcb mode histos are unusual, though, in that the hits in subpixels are added up int one LHCb pixel. instead of being just digital (per event) So 3 different subpixels hit increases bincount by 3 nstead of just 1.

This algorithm monitors the data-aquisition by checking the various header words, compares the information between the headers and verifies that the data can be read out correctly

AlertRate	number of events after which all alerts found in the past events are sent to the error reporting tool
MonitorRate	pre-scale for this algorithm, monitor only each nth event
SendAlertMessages	send alert messages to error reporting tool (true by default)
SendAlertDirect	do not accumulate alert messages but send them directly to the error reporting tool as they occour
OutputLevel	verbosity level

Used to create and apply masks for masking of hot pixels and testpatterns.

Here is a short description of how to use it:

1) run /group/rich/scripts/hotpixelmon.sh to produce the pixelmasks. They will be created as option files in the directory, where the script is run. Wait for at least 10000 Events and check, that the files make sense. The numbers describe pixels in a readable way: e.g. 2011431000 is Rich 2 Side 0 column 11 row 4 pixelcolumn 31 pixelrow 00 and subpixelrow 0

2) The files are produced every 1000 events and called HotPixelAB-NNNN.opts. A means Rich I (0) or II (1) B is A (0) or C(1) side. the NNNN denotes the number of Events, that have contributed to the file.

3) Copy the files HotPixel10* and HotPixels11* to cmtuser/Online_v3r3/Rich/RichMonitoringSys/v1r0/options/ and name them HotPixels10.opts and HotPixels11.opts

4) Now change the first line of both of these files to from HotPixels += { to HotPixels += { as the name of the aLHCb.Algorithm that reads them is SoftPixelMaskPost .

This algorithm is to be used in conjunction with the HPD test pattern that illuminates the four corner pixels. Produces a series of histograms that display only those four corner pixels as bins in order that each HPD can be visually inspected to ensure that it is sending data.

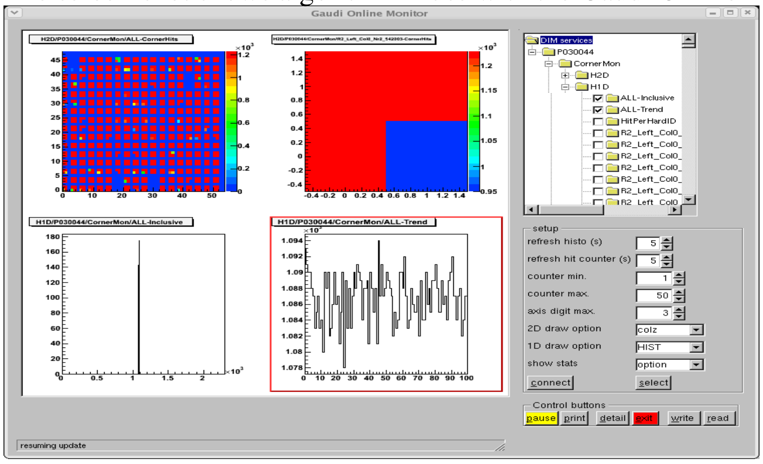
It publishes the following information via DIM

- as a summary
    - 2D histogram showing all corner pixels hits in all HPDs (in terms of pixel row and column)
    - 1D histogram of the inclusive distribution of the number of corner pixel hits of all HPDs
    - 1D histogram of the number of corner pixel hits of all HPDs as a trend plot
    - the number of corner pixel hits of all HPDs as a counter (real-valued number)
    - the moving average of the number of corner pixel hits of all HPDs as a counter (real-valued number)
- Per HPD
    - 2D histogram of the corner pixel hit distribution (in terms of pixel row and column)
    - 1D histogram of the inclusive distribution of the number of corner pixel hits
    - 1D histogram of the number of corner pixel hits as a trend plot
    - the number of corner pixel hits as a counter (real-valued number)
    - the moving average of the number of corner pixel hits as a counter (real-valued number)

The algorithm is controlled via the following `job options`

EventMoniRate	event rate to be monitored, e.g. 10 corresponds to monitoring only every 10th event
PurgeHistos	reset histograms after given number of events monitored
EventsPerFile	number of events after which the histograms are reset (if PurgeHistos = true)
NTrendSteps	number of entries in trend plot
histoCounterMin	lower (histogram border for inclusive hit distribution)
histoCounterMax	upper (histogram border for inclusive hit distribution)
MovingAverageEvents	number of monitored events over which the moving average is calculated
MonitorRich1Panel0	(true/false) monitor panel 0 of Rich1
MonitorRich1Panel1	(true/false) monitor panel 1 of Rich1
MonitorRich2Panel0	(true/false) monitor panel 0 of Rich2
MonitorRich2Panel1	(true/false) monitor panel 1 of Rich2
MonitorRich1Columns	vector of integers specifying the column in Rich1 to monitor
MonitorRich2Columns	vector of integers specifying the column in Rich2 to monitor
AliceMode	switch to ALICE mode for histograms

The screen-shot shows algorithm in use with the Gaudi Online Monitor



setup
refresh histo (s) 5
refresh hit counter (s) 5
counter min. 1
counter max. 50
axis digit max. 3
2D draw option colz
1D draw option HIST
show stats option
connect select

Control buttons
pause print detail exit write read

The displayed histograms are as follows:

Upper left histogram
: each group of 4 corner pixels for all HPDs in RICH2, where there is a 1 bin gap between each HPD;

upper right histogram
: an individual HPD and it can be seen that the bottom right corner pixel was not seeing the test pattern;

bottom left histogram
: total number of corner pixel hits that were seen per event;

bottom right trend
: same information as at the bottom left histogram, this time the total number of corner hits are plotted against the event number.

Monitoring using the special "Calibration" trigger

A special "calibration" trigger is foreseen which can be used by the various sub-detectors for dedicated tasks (e.g. flash a laser, etc). These events use a dedicated trigger and the resulting data is sent to the `CalibrationFarm` (~1 or 2 PCs) which analyse these special events. In contrast to other monitoring algorithms, the algorithms analysing these events operate on a dedicated single event and base their findings on this event only, whereas the other online monitoring events (in general) determine statistically significant quantities from multiple events. To obtain the relevant trigger information inside the Gaudi-based monitoring algorithm:

```
#include "Event/ODIN.h"
LHCb::ODIN* odin=get<LHCb::ODIN>(LHCb::ODINLocation::Default);
double triggerType = odin->triggerType() ;
```

You may also need to add to the options file:

```
EventClockSvc.EventTimeDecoder="OdinTimeDecoder";
```

The trigger types are:

3	periodic trigger
?	calibration trigger

Monitoring published histograms online

A first version of the official Presenter (currently `v0r10`) is provided by the Histogram & Monitoring-group in the control room. To invoke it:

- login to the `PLUS` cluster at IP8
- invoke `. /group/online/presenter/presenter.sh`

which then starts the official Presenter. You may need to edit the script and set the environment variable `DIM_DNS_NODE` to the appropriate value

How to test the software locally

The RICH online monitoring and data-quality application is called Panoptes To run a released version of the application, it is sufficient to

- Setup the software environment via `SetupProject --build-env Panoptes vMrN`
- When using it for the first time (or upgrading to new version), do the following:
    - Install the application via `getpack Rich/Panoptes vMrN`
    - Go to the `cmt` directory: `cd Rich/Panoptes/vMrN/cmt`
    - Configure the application: `cmt config`
    - Build the package: `cmt make`
- Set all environment variables: `source setup.(c)sh`

In some cases it may be necessary to install the latest version of the individual packages containing the monitoring algorithms and tools. In this case, get the corresponding package(s) via `getpack` your `cmtuser` area as well (and build them). Install the necessary packages (via `getpack`) in your `cmtuser` area.

Using the online environment

This setup uses the online software environment to test the monitoring algorithms locally including the interaction with DIM.

N.B. This part is currently changing as we move to a Python based configuration

Using the offline environment

To run the software offline:

- Setup Panoptes as described above
- Edit the file `Panptes-Default.py` in the `options` directory to configure the application (choose which algogrithms to run, etc)
- Edit the file `OfflineDataFiles.py` and add the files to analyse
- Run the application via `PanoptesOffline Panptes-Default.py`

Note

- When running at IP8, make sure that
    - the environment variable `DIM_DNS_NODE` does not point to a computer used by the online systems (e.g. `mona08`)
    - the line in `Panptes-Default.py` specifying the **Camera** server does not point to a computer used by the online systems (e.g. `r2ecs01`)
- The above description already refers to the new Python based options which is to released yet. It is available in IP8 by setting `User_release_area=~ukerzel/cmtuser_PanoptesDevel`. However, this will (hopefully) change soon.

RICH monitoring during LHCb data-taking

Setups

The software and options files are taken from the RICH area, i.e. `/group/rich/sw/cmtuser`. Therefore, extreme caution should be taken when working in this area as it is no longer exclusively used by the RICH group.

Scripts

The main script used by the global detector control FSM is `RichDAQMon.sh` in `/group/rich/scripts`.

The version of the RICH online monitoring used for the global data-taking is hard-coded in the script and has to be changed accordingly with each upgrade. It uses the script `setup.vars` in `Rich/RichMonitoringSys/vXrY/cmt` to setup the environment. This script is using `setup.sh` created by `cmt config` by executing `${CMTROOT}/mgr/cmt setup -sh -pack=RichMonitoringSys -version=v1r2 -path=/group/rich/sw/cmtuser/Online_v4r3/Rich -no_cleanup $* >` **N.B.** This has to be done for each release of the online monitoring software and when switching between "normal" and "debug" versions. **N.B.** All online systems use a 64-bit architecture now, make sure to build the online monitoring software for this environment

Further information

LHCb online monitoring project

The web-pages of the LHCb online monitoring group are found here

The meetings of the group are available via Indico, an (incomplete) list is collected below:

- 10 May 2006 Histograms and Monitoring
- 31 May 2006 Histograms and Monitoring
- 28 Aug 2006 Histograms and Monitoring
- 24 Oct 2006 Histograms and Monitoring
- 28 Nov 2006 Status report at LHCb week
- 8 Feb 2007 Histograms and Monitoring
- 13 Mar 2007 Status of the Online Presenter at the LHCb week
- 23 Mar 2007 Histograms and Monitoring
- 27 April 2007 Histograms and Monitoring
- 31 May 2007 Histograms and Monitoring
- 5 June 2007 Status report at LHCb week
- 28 Jun 2007 Histograms and Monitoring
- 04 Sep 2007 Histograms and Monitoring
- 11 Sep 2007 Online (LHCb week)

Hardware and L0 IDs for Rich2 C side

see attached file (provided by Antonis)

- Rich2_CSide_HardID_L0ID.txt: Rich2 C side, Hardware and L0 IDs

Data format

The EDMS document 768873 summarises the data format sent by the UKL1 boards

DIM

The online monitoring uses *DIM* (Delphi / Distributed Information Management system) to publish information and interact with the services. Further information can be found on the DIM webpages. Jump directly to the C++ documentation

The Online project is documented on these web pages

Publishing information

Information (histograms, counters, etc.) are published from the GaudiLHCb.Algorithm by calling `declareInfo`, the interface is defined by the IMonitorSvc class.

Clean up the event-builder

To make sure all tasks are ended, etc. the following command `sudo /group/online/scripts/cleaneb` kills all processes related to the event-builder

Misc

- StatEntity

Upgrade to a new release of the Online software environment

This paragraph summarises the steps needed to upgrade the online (monitoring) software environment to a new version of the Online release. Several packages are needed and multiple changes have to be made in the options files.

Main.ukerzel - 17 Mar 2007

- Rich2_CSide_HardID_L0ID.txt: Rich2 C side, Hardware and L0 IDs

---

This topic: LHCb/RichOperations > RichOnlineMonitor
Topic revision: r64 - 2008-09-09 - UlrichKerzel