Automated verification of BLM threshold changes

A. Skaugen for the BLM team

November 25, 2011

Overview of the BLM system

Main goal

Avoid quenches of the superconducting magnets and any damages induced by beam losses

Overview of the BLM system

Main goal

Avoid quenches of the superconducting magnets and any damages induced by beam losses

Layout

- About 4000 monitors situated in likely loss locations around the LHC ring
- The collected signals are integrated in 40μs time windows (via CFCs) and sent to the surface via redundant optical link
- The surface electronics continuously compares the losses to a set of predefined thresholds

Overview of the BLM system

Main goal

Avoid quenches of the superconducting magnets and any damages induced by beam losses

Layout

- About 4000 monitors situated in likely loss locations around the LHC ring
- The collected signals are integrated in 40μs time windows (via CFCs) and sent to the surface via redundant optical link
- The surface electronics continuously compares the losses to a set of predefined thresholds

Time and energy dependence

- The quench level of the magnets depend on the time scale of the loss and the current beam energy.
- Losses are integrated in 12 different running integration windows (running sums).
- The losses are compared to a set of predefined thresholds for each of the 12 running sums and 32 energy levels, for a total of $12 \cdot 32 = 384$ thresholds.

Overview of the BLM system

Main goal

Avoid quenches of the superconducting magnets and any damages induced by beam losses

Layout

- About 4000 monitors situated in likely loss locations around the LHC ring
- The collected signals are integrated in 40μs time windows (via CFCs) and sent to the surface via redundant optical link
- The surface electronics continuously compares the losses to a set of predefined thresholds

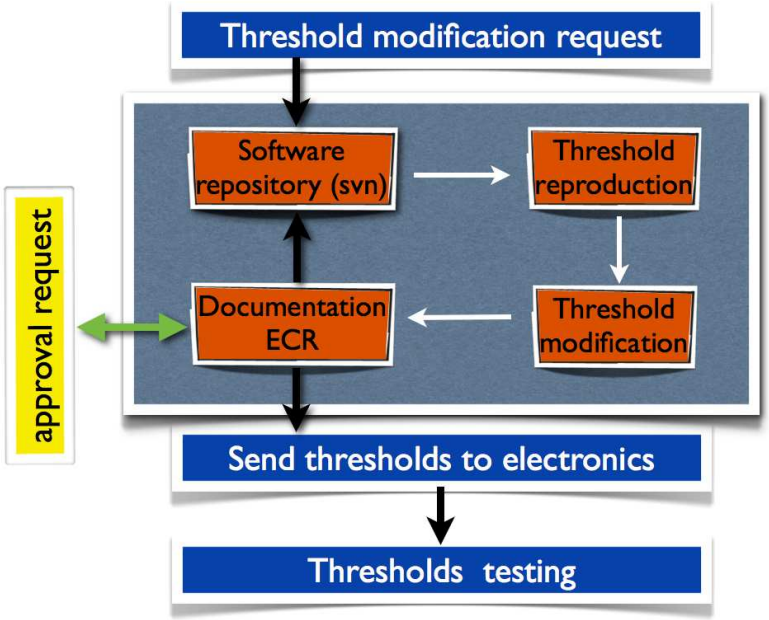
Time and energy dependence

- The quench level of the magnets depend on the time scale of the loss and the current beam energy.
- Losses are integrated in 12 different running integration windows (running sums).
- The losses are compared to a set of predefined thresholds for each of the 12 running sums and 32 energy levels, for a total of $12 \cdot 32 = 384$ thresholds.

Total number of thresholds

4000 detectors x 384 thresholds = 1.5 million

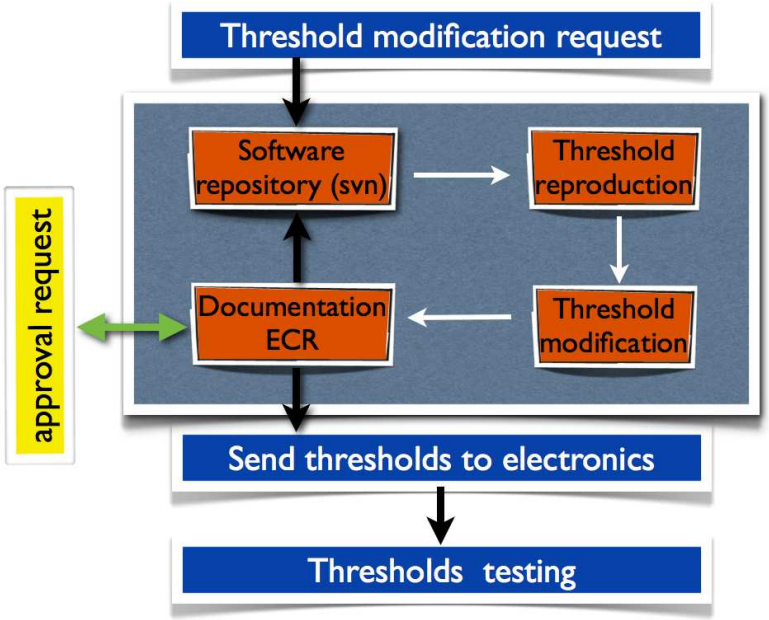
Motivation


Threshold modification request
Software repository (svn)
Threshold reproduction
Documentation ECR
Threshold modification
approval request
Send thresholds to electronics
Thresholds testing

Motivation

Existing tests

- A large number of tests are executed during the threshold deployment in order to minimize the probability of introducing wrong parameters into the system
- Most of the tests require visual verification and are susceptible to human error



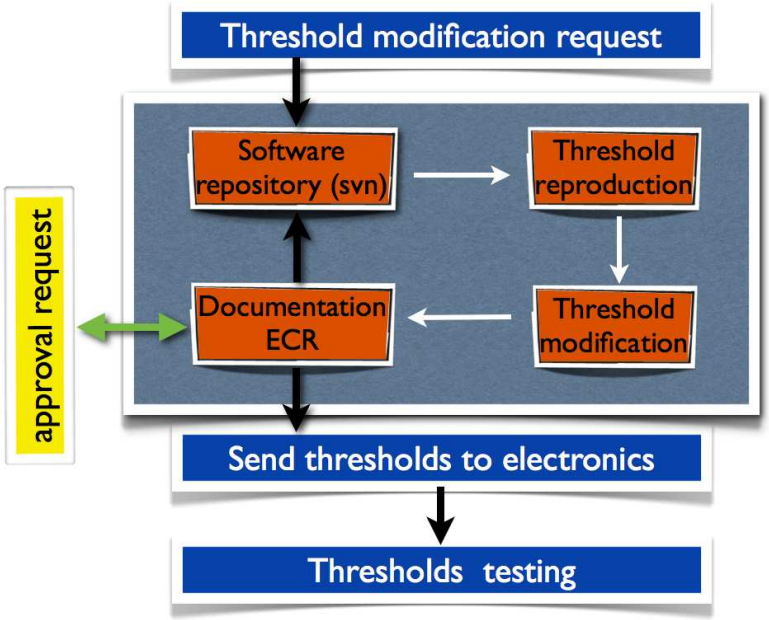
Motivation

Existing tests

- A large number of tests are executed during the threshold deployment in order to minimize the probability of introducing wrong parameters into the system
- Most of the tests require visual verification and are susceptible to human error

BLM audit

- “adequate protections against tempering or human errors are missing"
  “lack of software tools to detect erroneous values”
- The current efforts on threshold testing is being focused on automation



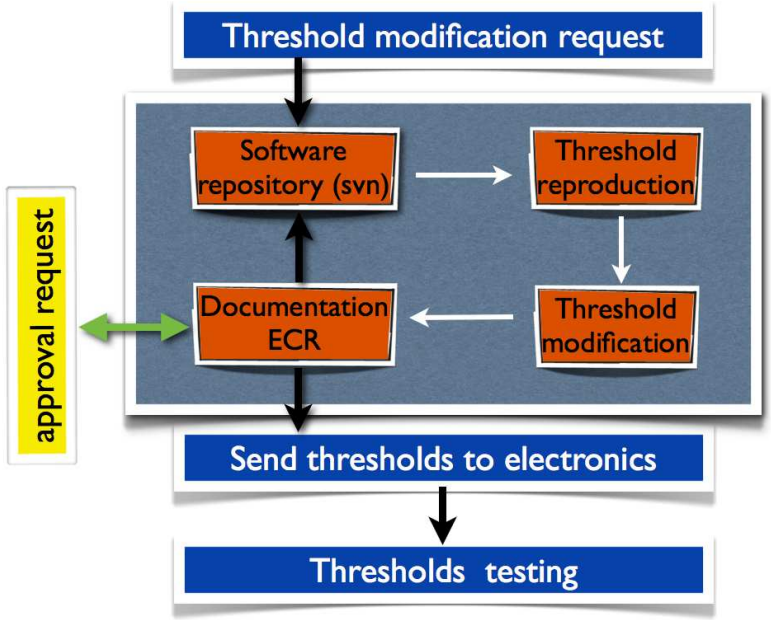
Motivation

Existing tests

- A large number of tests are executed during the threshold deployment in order to minimize the probability of introducing wrong parameters into the system
- Most of the tests require visual verification and are susceptible to human error

BLM audit

- “adequate protections against tempering or human errors are missing"
  “lack of software tools to detect erroneous values”
- The current efforts on threshold testing is being focused on automation



Requirements

- Recover the threshold table at any given time for all monitors around the ring
- Compare the recovered thresholds with each other
- Report any changes in a clear way

Recovering thresholds from the logging database

Logging database

- The BLM system sends the abort thresholds to the logging database
- Thresholds are logged only when they change
- A change can happen due to one of the following two reasons:
  - A change introduced by a user
  - In response to a change in the beam energy
- An MCS check is requested every time the thresholds are modified. During this check, the BLM crates go through all 32 energy levels.

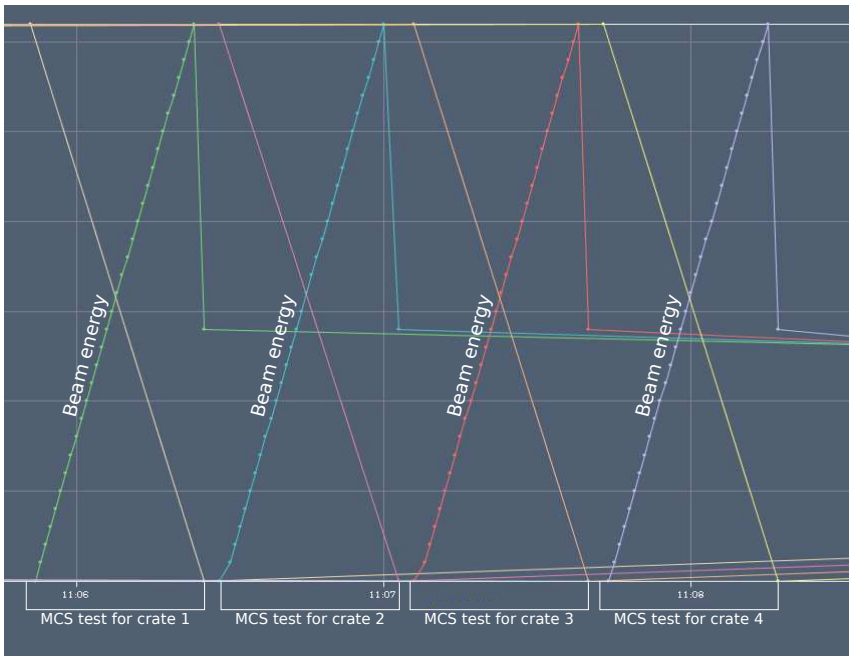
Recovering thresholds from the logging database

Logging database

- The BLM system sends the abort thresholds to the logging database
- Thresholds are logged only when they change
- A change can happen due to one of the following two reasons:
  - A change introduced by a user
  - In response to a change in the beam energy
- An MCS check is requested every time the thresholds are modified. During this check, the BLM crates go through all 32 energy levels.

Time difference

- The MCS tests happen at different times for different crates
- Beam energy is stored in a separate variable for each crate



Recovering thresholds from the logging database

Logging database

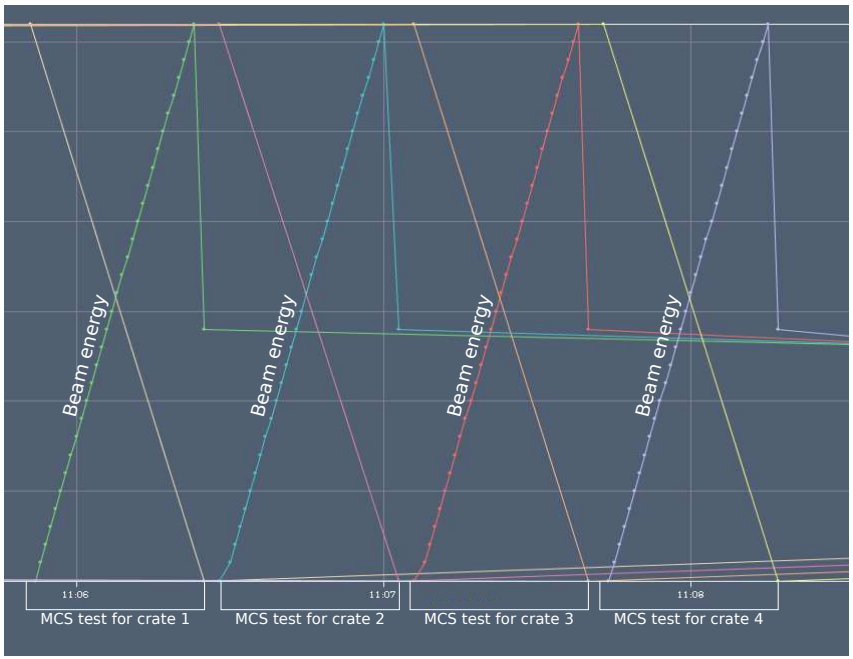
- The BLM system sends the abort thresholds to the logging database
- Thresholds are logged only when they change
- A change can happen due to one of the following two reasons:
  - A change introduced by a user
  - In response to a change in the beam energy
- An MCS check is requested every time the thresholds are modified. During this check, the BLM crates go through all 32 energy levels.

Time difference

- The MCS tests happen at different times for different crates
- Beam energy is stored in a separate variable for each crate

LSA db

The LSA db contains all the information needed to determine the mapping between monitors and crates.

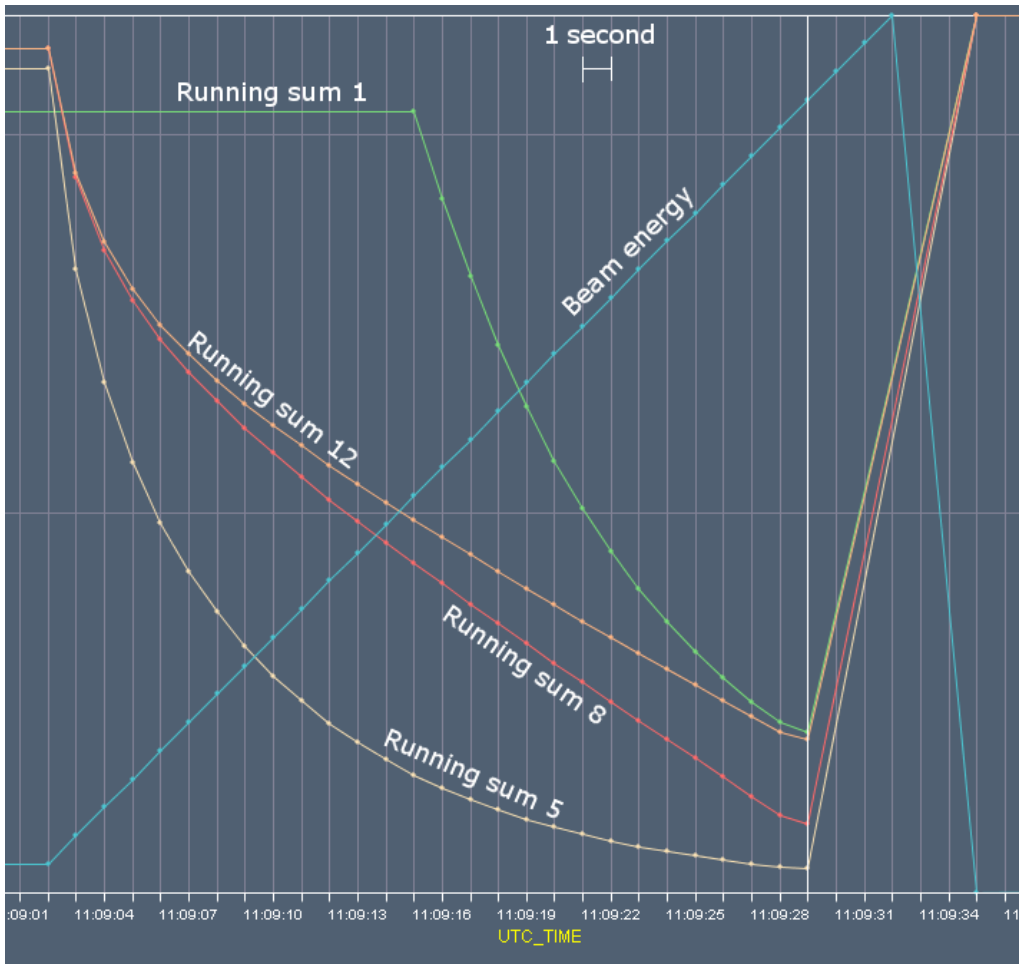


Threshold evolution during an MCS check

Logging of thresholds

- Thresholds are stored for each blm in one variable per running sum
- Thresholds are logged when they change. This happens when the beam energy changes.
  - Unless the thresholds are equal.

Example: `BLMQI.07L1.B2I10_MQ` (nov 13 2011)



Recovering threshold tables

Recovering the threshold table

1. The MCS checks are not simultaneous for the different crates. Therefore, we need to establish the mapping between monitors and crates.
   → LSA db

Recovering threshold tables

Recovering the threshold table

1. The MCS checks are not simultaneous for the different crates. Therefore, we need to establish the mapping between monitors and crates.
   → LSA db
2. For each crate:
   - Retrieve the `TEST_ACTIVE_CONSIST` flag, which indicates when there is an MCS check happening for a particular crate
   - Using this information, determine initial and final times of a MCS check

Recovering threshold tables

Recovering the threshold table

1. The MCS checks are not simultaneous for the different crates. Therefore, we need to establish the mapping between monitors and crates.
   → LSA db
2. For each crate:
   - Retrieve the `TEST_ACTIVE_CONSIST` flag, which indicates when there is an MCS check happening for a particular crate
   - Using this information, determine initial and final times of a MCS check
3. During each mcs check, retrieve energy levels and thresholds

Recovering threshold tables

Recovering the threshold table

1. The MCS checks are not simultaneous for the different crates. Therefore, we need to establish the mapping between monitors and crates.
   → LSA db
2. For each crate:
   - Retrieve the `TEST_ACTIVE_CONSIST` flag, which indicates when there is an MCS check happening for a particular crate
   - Using this information, determine initial and final times of a MCS check
3. During each mcs check, retrieve energy levels and thresholds
4. For each running sum:
   - For each logged change in energy level, recover the threshold logged simultaneously *or before*

Recovering threshold tables

Recovering the threshold table

1. The MCS checks are not simultaneous for the different crates. Therefore, we need to establish the mapping between monitors and crates.
   → LSA db
2. For each crate:
   - Retrieve the `TEST_ACTIVE_CONSIST` flag, which indicates when there is an MCS check happening for a particular crate
   - Using this information, determine initial and final times of a MCS check
3. During each mcs check, retrieve energy levels and thresholds
4. For each running sum:
   - For each logged change in energy level, recover the threshold logged simultaneously *or before*

Comparing thresholds

- The threshold tables for all monitors are recovered at two given times, using the above procedure
- For each monitor, energy level and running sum, the ratio between the new and old thresholds is computed

Command-line interface

The automated test has been implemented in Python with a command-line interface.

Command-line interface

The automated test has been implemented in Python with a command-line interface.

Look for changes between last two MCS checks

```
python LogDBChanges.py output/
```

Command-line interface

The automated test has been implemented in Python with a command-line interface.

Look for changes between last two MCS checks

```
python LogDBChanges.py output/
```

Changes between last two MCS checks happening before a given time

```
python LogDBChanges.py output/ -t "2011-10-13 14:00:00"
```

Command-line interface

The automated test has been implemented in Python with a command-line interface.

Look for changes between last two MCS checks

```
python LogDBChanges.py output/
```

Changes between last two MCS checks happening before a given time

```
python LogDBChanges.py output/ -t “2011-10-13 14:00:00”
```

Look between two specific times

```
python LogDBChanges.py output/ -t 2011-10-31 -t now
```

Command-line interface

The automated test has been implemented in Python with a command-line interface.

Look for changes between last two MCS checks

```
python LogDBChanges.py output/
```

Changes between last two MCS checks happening before a given time

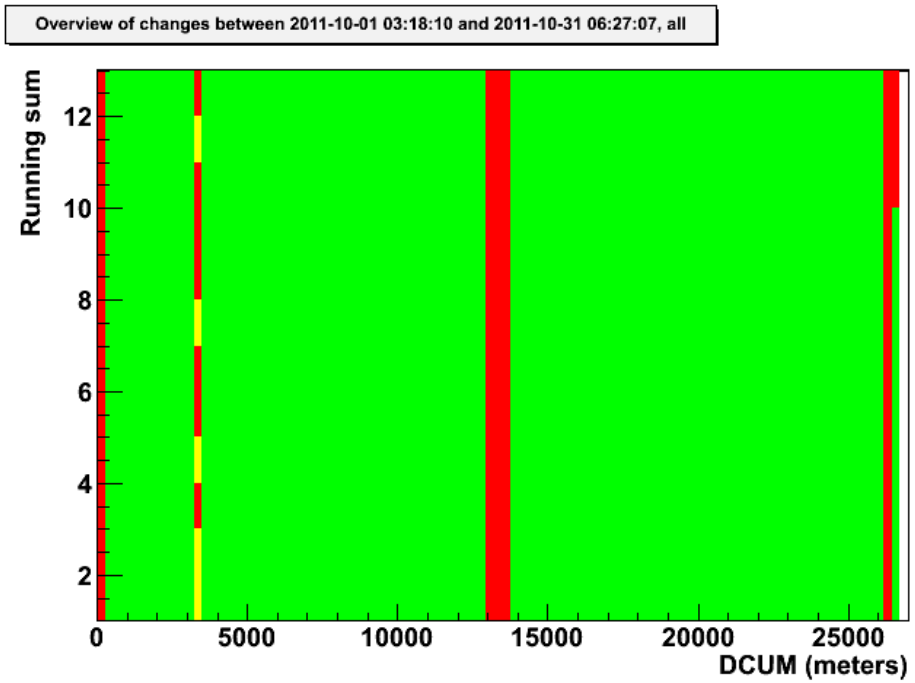
```
python LogDBChanges.py output/ -t “2011-10-13 14:00:00”
```

Look between two specific times

```
python LogDBChanges.py output/ -t 2011-10-31 -t now
```

Approximately 15 minutes running time. This makes it feasible to run the tool after every threshold modification and react quickly in case of undesired changes.

Overview histogram

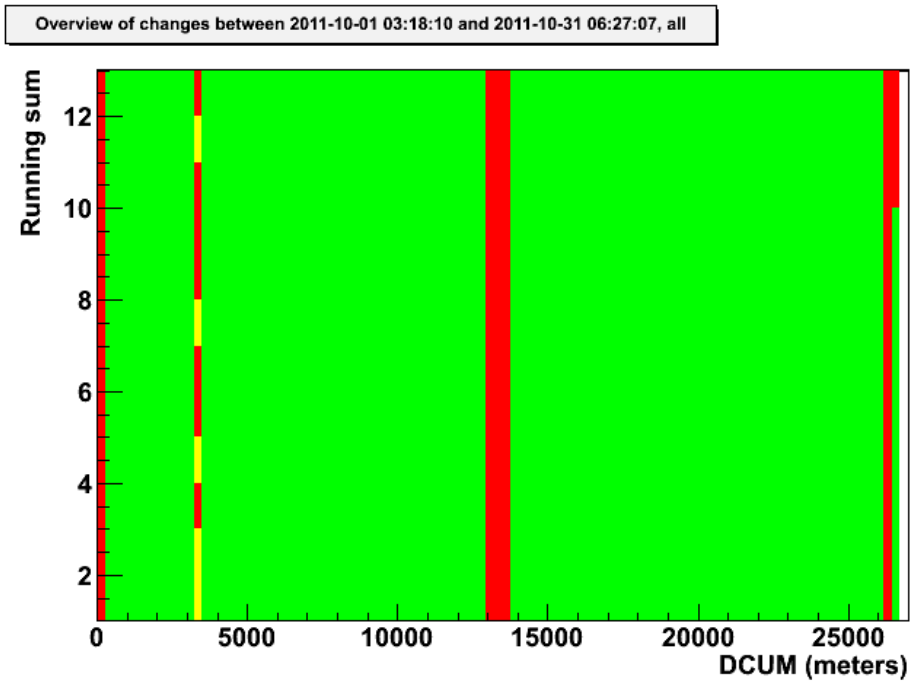


Green → No change

Yellow → Small change

Red → Large change

Overview histogram



Green → No change

Yellow → Small change

Red → Large change

Definition of size

$$x = \sqrt[32]{\prod_{\text{Energy levels}} \frac{\text{new threshold}}{\text{old threshold}}}$$

$x < 1.0001$: No change

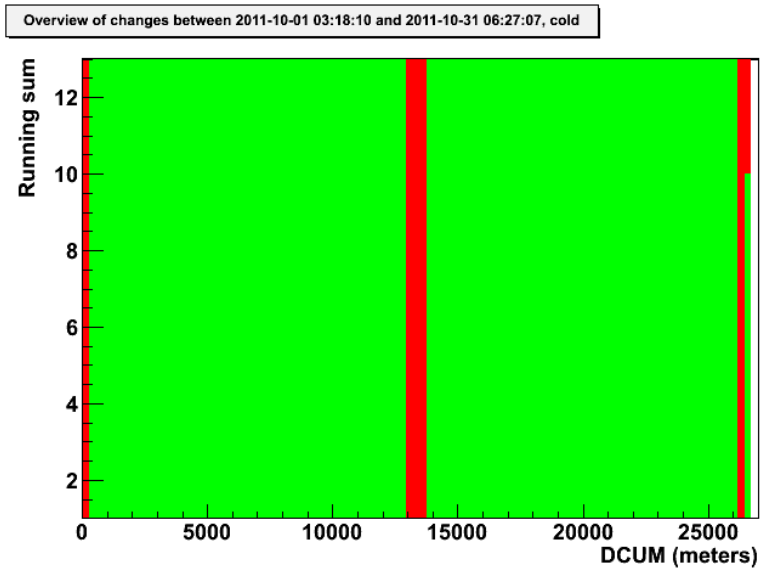
$1.0001 \leq x < 2$: Small change

$x \geq 2$: Big change

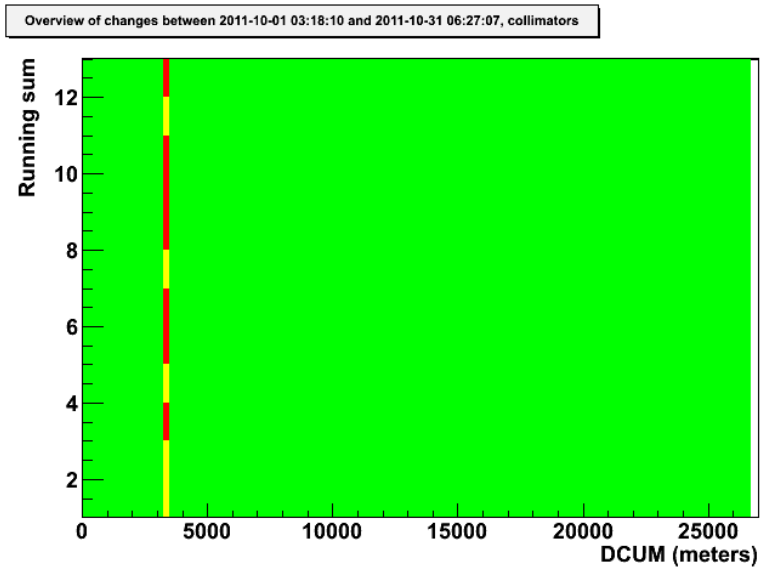
Categories

Plots are also split depending on what kind of element the monitor protects

Cold magnets:

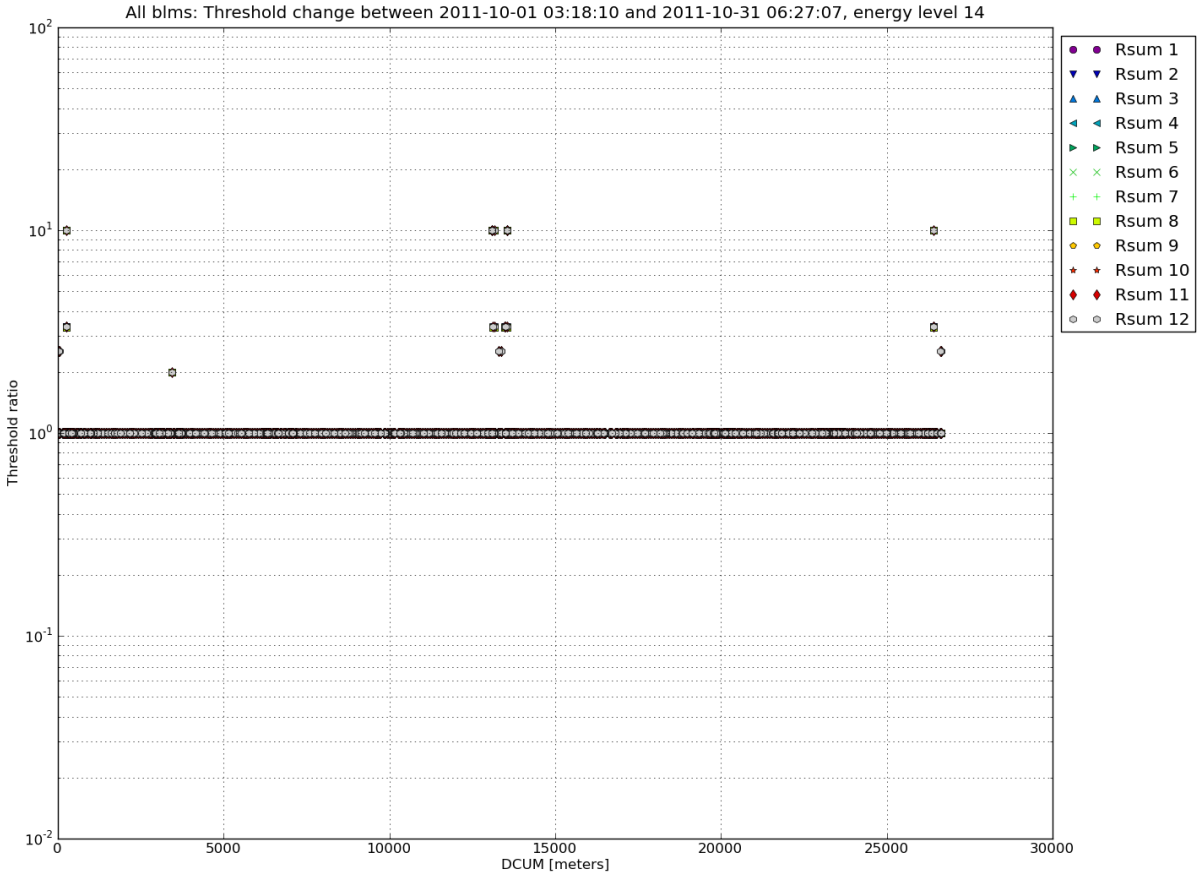


Collimators:



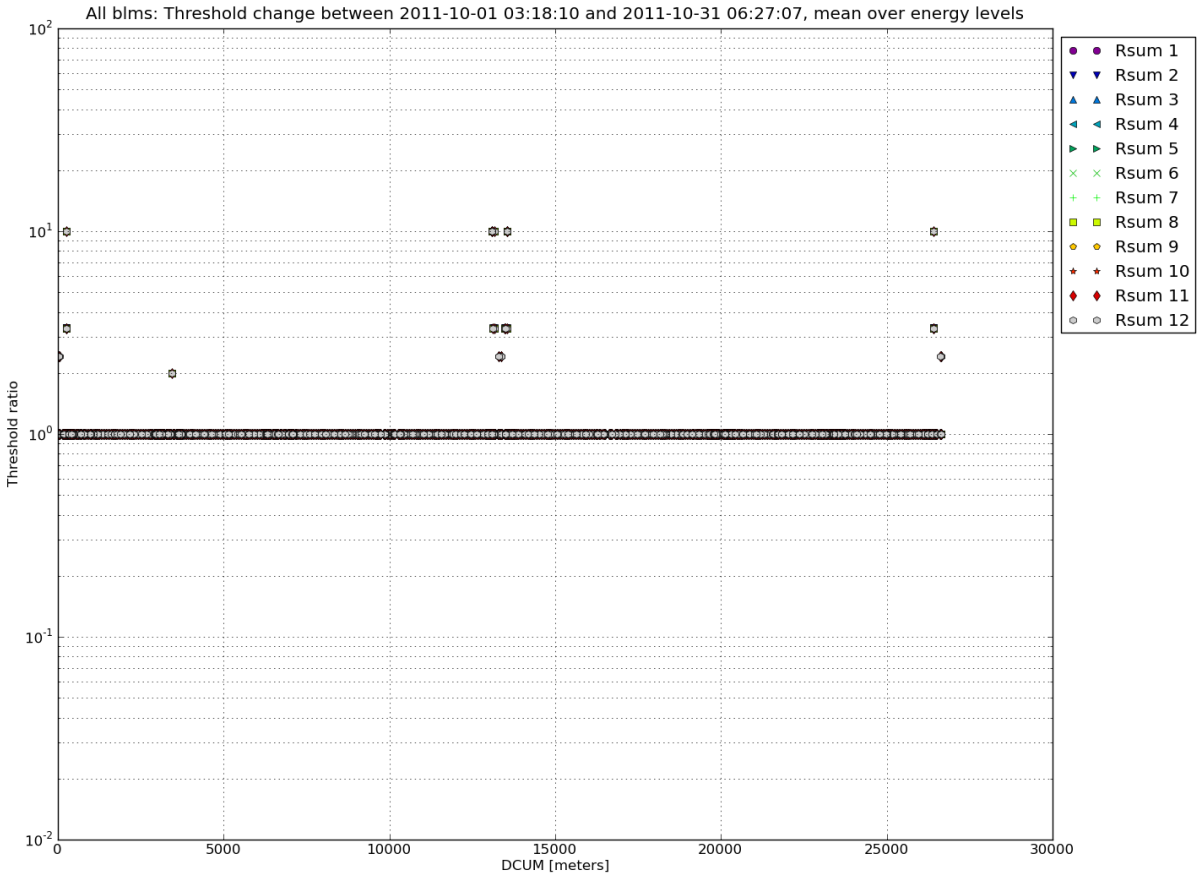
Energy levels

Plot the ratio of change vs. dcum for one particular energy level



Mean of all

Same as above, but take the geometric mean over energy levels



Textual output

```
Family THRI.SS.B1.1_MQM:
    BLMQI.06R5.B1E10_MQML_XRP, crate CFV-SR5-BLMR, dcum 13556:
        Change between 2011-10-01 03:17:32 and 2011-10-31 06:25:15:
            All energy levels:
                All running sums changed with ratio 10

Family THRI_2_MQXB_LumLoss:
    BLMQI.02R1.B2I23_MQXB, crate CFV-SR1-BLMC, dcum 34:
        Change between 2011-10-01 03:16:55 and 2011-10-31 06:26:30:
            Energy levels [0, 1]:
                Running sums [10, 11, 12] changed with ratio 2.63757
            Energy level 2:
                Running sums [10, 11, 12] changed with ratio 2.63229
            Energy level 3:
                Running sums [10, 11, 12] changed with ratio 2.62806
            Energy level 4:
                Running sums [10, 11, 12] changed with ratio 2.62301
            (...)

Family THRI_XRP:
    BLMEI.07L1.B2E20_XRP, crate CFV-SR1-BLML, dcum 26402:
        Change between 2011-10-01 03:18:10 and 2011-10-31 06:27:07:
            All energy levels:
                All running sums changed with ratio 3.33381
```

Cronjob

- The tool was designed to be run after every threshold deployment to prodect against erronous values
- We also get the ability to run the tool automatically at regular intervals, in order to keep a history of expected and unexpected changes
- It currently runs on a linux machine once per week, with reference dates the execution time and one week before
- When done, a report is sent via email to a list of BLM experts

Conclusion

- A tool has been developed for threshold verification.
  - Allows to recover threshold tables from the logging database at a given time for all monitors around the ring
  - Performs comparisons between two sets of threshold tables monitor by monitor
- Step forward on the automation of thresholds test and verification
- Tool executed after thresholds modification
- Executed automatically on a weekly basis (since October 31st)

Vectornumeric

The variables

- New logging format starting from September
- 12 new variables:
  - `LHC.BLMI:THRESH_RS01`
  - `LHC.BLMI:THRESH_RS02`
  - ...
  - `LHC.BLMI:THRESH_RS12`
- Logs thresholds for *all* BLMs

The advantage

Tens of times faster database lookup when interested in all BLMs