BRMSMon Status Update

Nicolas ARNAUD (narnaud@lal.in2p3.fr)

Detchar meeting, 2015/10/30

BRMSMon Package

- Developed by Florent and used during the Virgo data taking periods
  - Updated for AdV (currently v2r1)

- SVN: Repository, Journal of the revisions, RSS flux

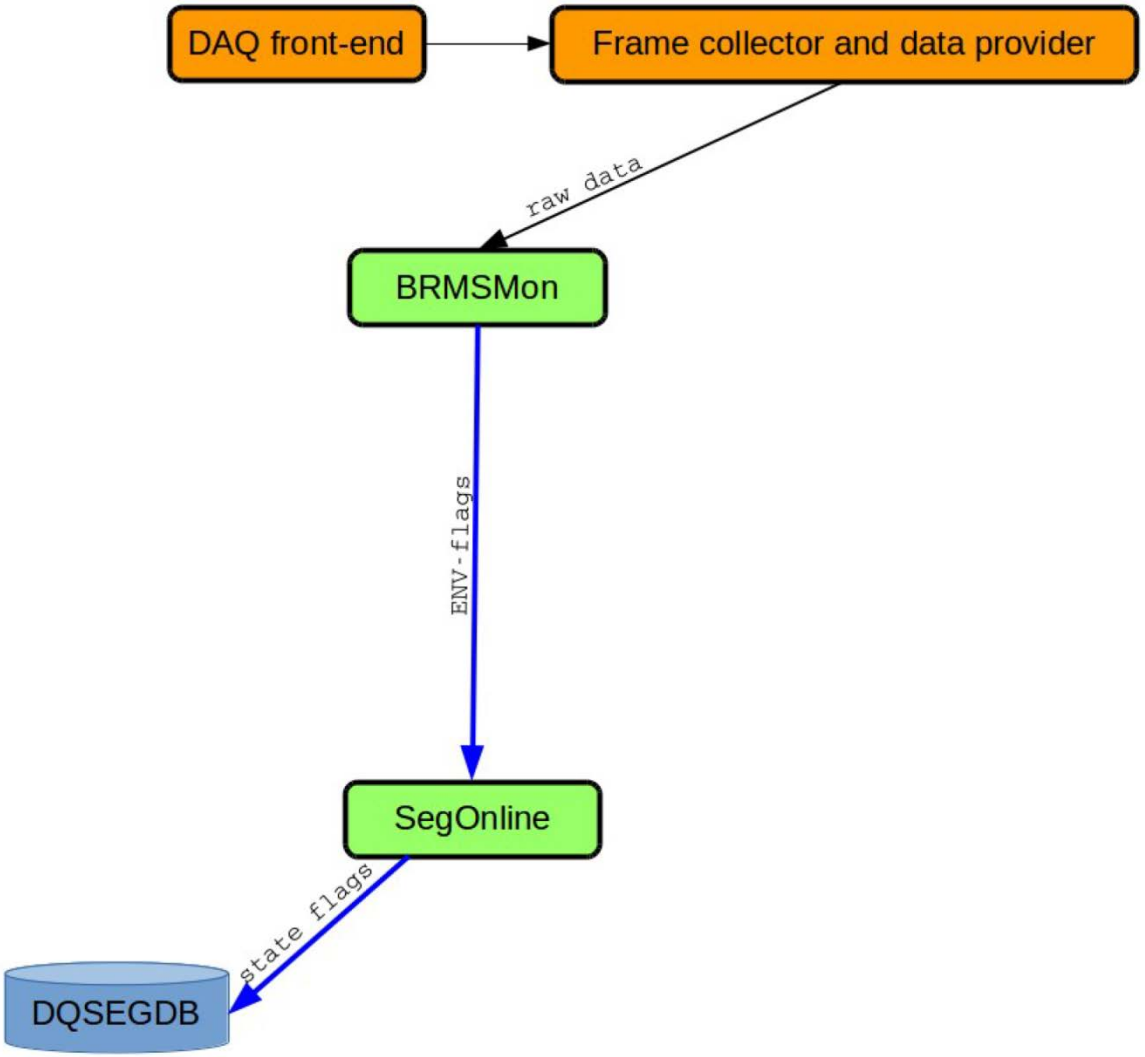
- Documentation (updated, with reference to the old documentation)

- Running online using VPM
  - Currently not from the official Virgo software area – but from my user account
  - Configuration file: /virgoData/VirgoOnline/BRMSMon.cfg
  - Frames produced by BRMSMon are processed by SegOnline (thanks to Didier) and appear in the raw data

- BRMSMon channels have V1:BRMSMon as prefix
  - Updated at 1 Hz – using 1 second-long frame as input

Online architecture

- From Florent’s slides



BRMSMon algorithm

- Produces DQ flags based on band-RMS computed from input channels
- Currently focuses on environmental channels
  - No check of the IFO lock status (was different in the Virgo era)
  - In the future: DQ flags delivered or not depending on an external flag (SCIENCE)
- DQ flag definition

```
DQ_NAME dq_name  n_coinc
BRMS_CHANNEL chan_name_1  fft_length_1  f_min_1  f_max_1  f_max_1  n_sigma_1  cycle_1
BRMS_CHANNEL chan_name_2  fft_length_2  f_min_2  f_max_2  f_max_2  n_sigma_2  cycle_2
...
BRMS_CHANNEL chan_name_N  fft_length_N  f_min_N  f_max_N  f_max_N  n_sigma_N  cycle_N
```

where:

```
dq_name    : Name of the DQ flag
n_coinc    : At least n_coinc Band-RMS out of N must be above threshold at the same time to set
the DQ flag at 1.
chan_name  : Channel name
fft_length: Time length for the FFT (sec)
f_min      : Lower boundary of the frequency band (Hz)
f_max      : Upper boundary of the frequency band (Hz)
n_sigma    : Threshold (adaptive or static) see the threshold section for details.
cycle      : The threshold is adapted every cycle second. This option is irrelevant for a
static threshold.
```

  - DQ flag set when n_coinc band-RMS out of N are above threshold
  - For each input channel, set the FFT length (in s), the frequency range [f_{min};f_{max}], decide whether the threshold to be applied is static or adaptative, at what level the threshold should be, and how often it will be updated (if adaptative)

BRMSMon algorithm (cont'd)

- Static or adaptative threshold

```
BRMS CHANNEL chan name 1  fft length 1  f min 1  f max 1  f max 1  n sigma 1  cycle 1
```

  - if $n_{sigma} < 0$, the threshold is static and threshold = $| n_{sigma} |$
  - if $n_{sigma} > 0$, the threshold is adaptative and defined in the following way:
    threshold = mean(band-RMS) + n_{sigma} × rms(band-RMS)
    In that case, it is updated every cycle seconds

- The mean and rms values of the band-RMS distribution are usually computed from the band-RMS values which are below the current threshold
  - We want to see glitches which are significantly above the normal range of variations of that particular channel

- Yet, if the channel output changes significantly and « permanently », BRMSMon should follow this evolution
  - If more than 75% of the band-RMS values are above the current threshold, the updated threshold is computing using only the values above the current threshold

BRMSMon algorithm (cont'd)

- For channels with adaptative thresholds, the first (meaningful) threshold is computed based on the first 60 seconds of data
  - No DQ flag is set during this first minute
- The adaptative thresholds are then updated again 5 minutes later, and then every cycle seconds – see previous slides for the algorithm
- BRMSMon checks for each channel
  - that the FrVect* pointer returned by FrameFindVect is not NULL
    → Identify missing channels
  - that the computed band-RMS is not 0
    → Identity dead channels!?
- Each DQ flag produces 2 × (number of input channels) + 1 channels in the raw frame
  - V1:BRMSMon_<DQ name>
  - V1:BRMSMon_<DQ name>_V1:<channel_1 name>_<f_{min1}>_<f_{max1}>Hz_rms
  - V1:BRMSMon_<DQ name>_V1:<channel_1 name>_<f_{min1}>_<f_{max1}>Hz_thr
  - V1:BRMSMon_<DQ name>_V1:<channel_2 name>_<f_{min2}>_<f_{max2}>Hz_rms
  - V1:BRMSMon_<DQ name>_V1:<channel_2 name>_<f_{min2}>_<f_{max2}>Hz_thr
  - (…)

Configuration file

- Currently includes all channels pointed out by Irene
- But no (real) attempt yet to define meaningful DQ flags based on these input channels

→ Time to be creative!

- Some examples from the Virgo era .cfg file listed in the next slide

DQ flag examples

- Same channel, low/medium/high thresholds

```
DQ_NAME    RECYCLING_GLITCH_LOW  1
BRMS_CHANNEL    Gc_Recycling      2   20   30    6   3600

DQ_NAME    RECYCLING_GLITCH_MID  1
BRMS_CHANNEL    Gc_Recycling      2   20   30    8   3600

DQ_NAME    RECYCLING_GLITCH_HIGH 1
BRMS_CHANNEL    Gc_Recycling      2   20   30   10   3600
```

- Select frequency bands

```
DQ_NAME    TCS_NI_POWER_GLITCH    1
BRMS_CHANNEL    TCS_NI_Power    0.1    32    64    10   500
BRMS_CHANNEL    TCS_NI_Power    0.1   256   512    10   500
```

- Split a frequency band into several frequency ranges

```
DQ_NAME    B5_LOW_FREQ_TIGHT    1
BRMS_CHANNEL    Pr_B5_ACq    1      4    16    6   3600
BRMS_CHANNEL    Pr_B5_ACq    1      8    32    6   3600
BRMS_CHANNEL    Pr_B5_ACq    1     16   128    6   3600
BRMS_CHANNEL    Pr_B5_ACq    1    128  1024    6   3600
```

- Different DQ flags for each frequency band of a given channel

```
DQ_NAME    AC_NE_64_128    1
BRMS_CHANNEL    Em_ACBDNE01    0.1    64    128    9   10000
DQ_NAME    AC_NE_128_256    1
BRMS_CHANNEL    Em_ACBDNE01    0.1   128    256    9   10000
DQ_NAME    AC_NE_256_512    1
BRMS_CHANNEL    Em_ACBDNE01    0.1   256    512    9   10000
```

→ Would location-related DQ flags make sense?

- That would mean mixing apples and oranges: e.g. seismic sensors and microphones