Virgo DQ and vetoes for VSR2/S6 online analysis

The Virgo Data Quality group

Version 4.3
November 18, 2008

1. Introduction

The motivations for real-time analyses during VSR2/S6 are described in [1] for the burst search and in [2] for the inspiral search. A very fast answer from gravitational wave (GW) detectors would allow electromagnetic follow-up of candidates and respond promptly to external triggers. Online analyses will be performed both on detector site and in a centralized place where data will be transferred online. Data Quality (DQ) and vetoes availability will be fundamental to eliminate fake candidates and to assure quality and reliability to the online analyses. Fig. 1 shows a scheme of the proposal done by the burst group for the online analysis of S6 and VSR2 data. Fig.2 shows the current scheme proposed by the burst group how DQ and vetoes segments will be generated and used. The scheme is meant for any detector (H1, H2, L1 and V1).

Add here CBC online analysis scheme (online analysis done with a maximum latency of 1 week. + single ITF triggers generation?)

The goal of this document is to describe what the Virgo collaboration plans to do for online VSR2 data quality assessment. It mainly deals with the generation of online DQ information but mentions as well the event by event vetoes generation. It also describes briefly the data quality tools and checks that are going to be used for VSR2 in the Virgo control room.

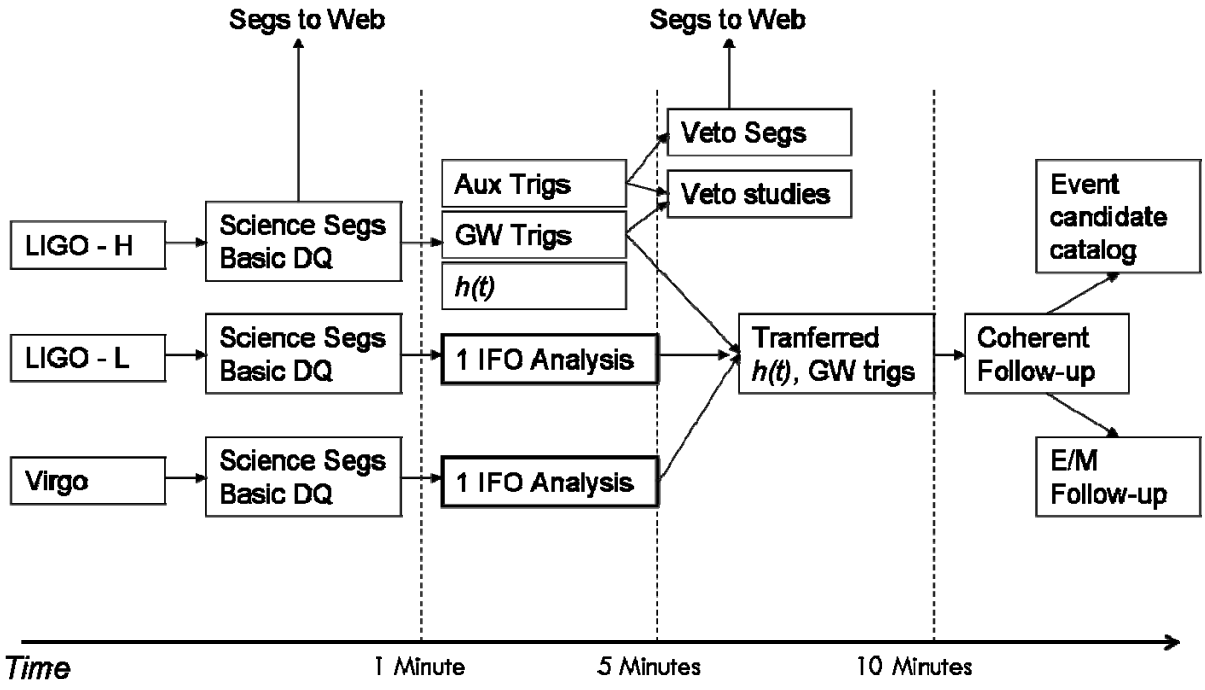

Fig. 1: Diagram of the burst online analysis proposal [1].

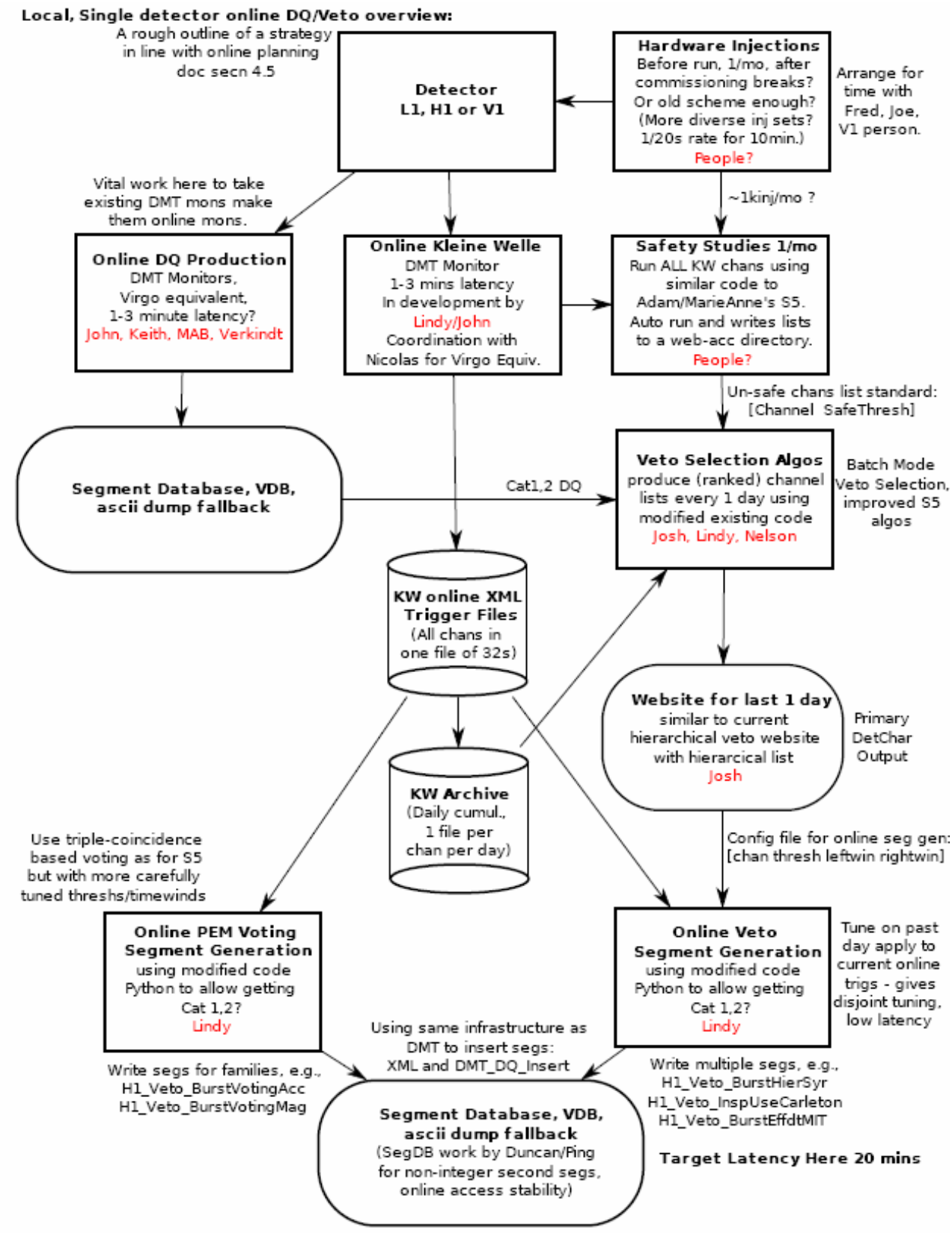

Fig. 2: Scheme of current online DQ/Vetoes for S6/VSR2

2. Requirements

For any analysis, single instrument based or coherent analysis, trigger generation requires ITF_SCIENCEMODE segments. One could also use the ITF_LOCK segments (which corresponds to the achievement of the final step of the locking procedure). Besides, some search pipelines are going to be fully integrated in the Virgo data acquisition system (DAQ). They will thus receive data online from the h(t) reconstruction. They will also receive some data quality information (cat I & II) that can be defined at that level (see Fig. 3 and section 6.1 for more details).

2.1 Latency requirements

Basic segments like ITF_SCIENCEMODE and ITF_LOCK segments must be generated automatically with almost no latency (well below one minute). The other DQ segments can be generated with greater latency, from few minutes up to a few hours, depending on the category and the type of searches (burst or inspiral) [1]. One expects, for instance that cat I DQ flags are generated within a minute.

2.2 Reliability requirements

In addition to these latency requirements, the DQ segments must be produced in a reliable way (no missing information). This requires a reprocessing scheme that allows to complete and validate DQ segments in case of missing data or DQ monitors crash (see section 6.4).

2.3 Storage and access

The DQ segments must be stored in a database (VDB). Some ASCII files might be as well generated but VDB will be the official DQ segments repository.

2.4 Veto list production

The DQ flags cannot cover all sources of glitches that affect the analyses. Some vetoes based on transient noise triggers must be developed as for VSR1, but with a rather low latency. We hence need to be able to process all the Virgo ancillary channels online. A fast glitch finder algorithm is required (KW). Veto algorithms to determine which channels are relevant to define veto list will have to be run just after the triggers production. Common strategy with LSC will be applied in order to have the same definition of vetoes. Finally, the safety of any retained channel must be assessed.

2.5 Computing and storage requirements on site

Computing:

- One machine “a la olserver14” to feed online the VDB with basic segments
- At least one machine “a la olserver14” to compute online all the category I and category II DQ flags. This machine is already the additional one requested for online noise monitoring.
- (20+N) machines to run in-time the algorithms for online vetoes production (20 for KW and N for the other algorithms).

Storage:
- A minimum of 10 GB in the web area to display online DQ results and to archive them.
- KW triggers storage: 2.5 GB / month
- Q triggers storage:
- Other triggers (WDF, outlierMoni, VirgoHACR…) storage: a buffer of 1TB is already requested for online noise monitoring. In addition, some algorithms like VirgoHACR and WDF may use a mySQL database whose requirements are TBD.

2.6 Deliverables

- "Basic" segments (ITF_SCIENCEMODE, ITF_LOCK, and INJECTIONS) online generation
- Automatic generation of "standard" DQ that are known to be useful and well defined (see list in section 3).
- Although the goal of this document is to describe the online DQ and vetoes generation, one should remind that it does not prevent to define offline DQ flags that require a deeper understanding of the data. Such DQ flags are then used to build the DQ lists that will be used by offline analyses. These developments may be also the first step of an online implementation.
- For the veto list production, the channels and thresholds will be defined before the run. This concerns vetoes based on KW triggers and any other vetoes that will be developed

3 List of online DQ flags

During VSR1, DQ flags have been identified that are known to be useful for the analyses. Some of them (category I and II listed below) are planned to be produced online in VSR2. The responsible column indicates who is responsible to do the online implementation and to perform all the checks before and during the run. This list is not finalized and should be completed by new DQ ideas, especially after the first weeks of the run.

Some DQ flags in this list do not need for new monitors to be produced because they already can be output by the online detector monitoring system of Virgo. They will then be collected according to the scheme described in Fig. 3, section 6. Those flags are marked below with a (DM) in the "comments" column.
[Didier: this is to be discussed. Simple monitors could be within detector monitoring system but developing a specific monitor (outside detector monitoring) is more flexible if a complexification is foreseen like using combination of channels (like for instance the flag GROUND_50HZ).]

Flag name	**Description**	**Comments**	**Responsible**
V1:ITF_LOCK	Virgo reaches the last step (12) of the locking procedure	**(DM)**	D. Verkindt
V1:ITF_SCIENCEMODE	Virgo state is "Science"		D. Verkindt
V1:INJECTION_INSPIRAL	Inspiral hardware injection periods	**(DM)**	
V1:INJECTION_BURST	Burst hardware injection periods	**(DM)**	
V1:INJECTION_PULSAR	Pulsar hardware injection periods	**(DM)**	
V1:INJECTION_STOCHASTIC	Stochastic bkgd hardware injection periods	**(DM)**	

V1:DARKF_MISS	Identifies periods where Pr_B1_ACp channel is missing	**(DM)** Latency: < 1mn (last segment opened)	D. Verkindt
V1_RAWFRAME_MISS	Identifies periods where raw frame is missing	Latency: < 1mn (last segment opened)	D. Verkindt
V1:HREC_MISS_ONLINE	Identifies periods where online h(t) channels are missing	Latency: < 1mn (last segment opened)	D. Verkindt
V1:HREC_BADQUALITY_ONLINE	Identifies periods where online h(t) reconstruction is generating bad quality h(t) (calibration lines not visible, ….)	Latency: < 1mn (last segment opened)	D. Verkindt
V1:MAINTENANCE	Identifies periods where ITF is in science mode while fixings are done on it.	Latency < 5mn (last segment opened)	

Flag name	**Description**	**Comments**	**Responsible**
V1:PRE_LOCKLOSS_10S	Identifies the last 10 S before ITF unlocks		
V1:ITF_LOCKED_FOR_LESS_THAN_300S	Identifies the periods where the ITF has reached its state SCIENCE since less than 300 s.		
V1: SAT_XXX	Identifies periods where some channels playing an important role in the control saturates. Example: V1:SAT_COIL_NEWE, V1:SAT_SSFS_CORR, Etc ….	**(DM)** Channels and thresholds to be chosen in advance (same as for VSR1?)	
V1:SAT_COIL_NEWE	Combines the saturation DQ flags generated by Hrec using the 4 Sc_NE/WE_RM_CoilU/D channels.	Like for VSR1 a merge using a window 10 s should be applied …	MAB
V1:SAT_CA_COIL_NEWE ????	Same as V1:SAT_COIL_NEWE, but using the Ca_XX channels	Do we need it?	
V1:SAT_SSFS_CORR	Use the saturation check performed by Hrec of the Sc_IB_SSFS_Corr channel	Follow VSR1 recommendation	MAB
V1:SAT_D2_ACQ	Combines the saturation DQ flags generated by Hrec on the channels Pr_B1_d2_ACq	Do we need to do the same for d3?	MAB
V1: BRMS_XXX	Identifies periods where BRMS on some channel is above a chosen threshold	**(DM)** Channels and thresholds to be chosen in advance (new channels wrt VSR1?)	

Flag name	**Description**	**Comments**	**Responsible**
V1: GROUND_50HZ	Identifies periods where power line glitches (50Hz) generates glitches in the dark fringe	Based on Em_MABDXX channels glitches asking for coincidence between different location (NE, WE, CE)	MAB
V1 : EARTHQUAKE	Identifies periods where	Based on V1:Em_SEBDCE04_rms	

	earthquake generates dark fringe noise increase if not unlocks	(to be checked if this is the best). Thresholds to tune wrt VSR1 thresholds	
V1: AIRPLANE	Identifies periods where airplanes generates acoustic/seismic noise that couples with the dark fringe	Channels and thresholds to determine (use of VSR1 studies?)	
GLITCH_RATE_OVERXPERMN_THY	Identifies periods where the glitchiness of Pr_B1_ACp goes above X events per minute over a threshold Y.	Based on results of VirgoHACR, WDF, outlierMoni or Qonline over the full frequency bandwidth of Pr_B1_ACp	
GLITCH_RATE_OVERXPERMN_THY _FMINHZ_FMAXHZ	Identifies periods where the glitchiness of Pr_B1_ACp goes above X events per minute over a threshold Y, between FMIN and FMAX	Based on results of VirgoHACR, WDF, or Qonline over a specified frequency bandwidth of Pr_B1_ACp	

4. List of online veto channels

During VSR1, a list of auxiliary channels that may be useful to produce vetoes for the analyses, has been identified. During VSR2, each of those channels will be processed online by the KW algorithm. During commissioning and during the first weeks of VSR2, threshold and frequency range will be tuned and new channels may be added to this list.

Channel name	**Description**	**Comments**
All Em_*	Environment monitoring channels	Only low frequencies (f<600 Hz) for magnetic probes
A selection of Pr_*	Photodiodes channels not used in control loops	
A selection of Sc_*	Correction signals of alignment control loops	
A selection of Bs_*	Input beam control signals	
A selection of Ca_*	Coils current used by calibration signals	Most of them are “unsafe”

5. VDB: the DQ segments main repository

An important piece of the Virgo infrastructure concerning the data quality is the Virgo Data Base (VDB) [8] that is the main DQ segments repository. The VDB package comes with a C and python library that can be used to put and get segment lists directly to or from the database.

The VDB package contains also VDB_SegList.exe that can feed online VDB by reading the Virgo trend data (latency=30mn). This tool can be used for a first level reprocessing of the DQ segments produced online (see 5.2). An other tool, SegOnline, is developed specifically to feed online VDB with a latency below 1mn (see 6.1).

The DQ segments produced online will be stored in VDB with a version number (TBD) specific for the online DQ segments, in order to keep track of what has been produced online. The access to the DQ segments in VDB is possible from:

- a web page interface **http://vdb.virgo.infn.it/main.php**
- a command line toolkit **VDBtk_segment.exe** available in the VDB package available in the Virgo CVS. The VDB package comes with a C and python library that can be used to upload and download segment lists directly to or from the database. The VDBtk_segments.exe can be used to provide online segments lists combination and to retrieve for some significative event information about DQ lists active at a given GPS time window.

6. Online DQ production, storage and access

The Virgo data acquisition (DAQ) has been developed since years. It especially already includes modules/interface that allow to process on-fly data coming from the detector [6-7].
Among them, one should quote the online h(t) process and the online detector monitoring. These 2 “boxes”, already integrated in the DAQ chain, are important since they generate Data Quality information that are used to define DQ segments. In order to make the DQ segments production automatic, we will need to introduce some modifications and develop interfaces with the main DQ segments repository (VDB).
Moreover, online pipelines (MBTA, Merlino) have already been interfaced with the online h(t) production system. They receive directly h(t) vectors and some data quality information generated by the h(t) process. At that level, the data quality information is stored in some FrSerData structure in the same frame that contains h(t). The same data quality information will be used to define the corresponding DQ segment lists.

6.1 Implementation scheme

As shown on Fig. 3, the proposal is to use the DAQ tools [6-7] already implemented at the level of the online h reconstruction (HRec) and the online noise monitoring (FbmNoise) to feed the monitors that will produce the online DQ flags. Those flags (including those produced by the h(t) reconstruction process) are embedded in frames and passed to a second stage FbmDQ where the process SegOnline builds the DQ segments and feed online the Virgo Data Base (VDB).
In parallel, a second process SegToDAQ sends the DQ flags to DAQ, allowing them to be stored in the trend data.

If needed, SegOnline will produce also ASCII files containing the segments lists (one file per DQ flag) and updated every 10s. Each file will be located in /opt/w3/online/dq/.

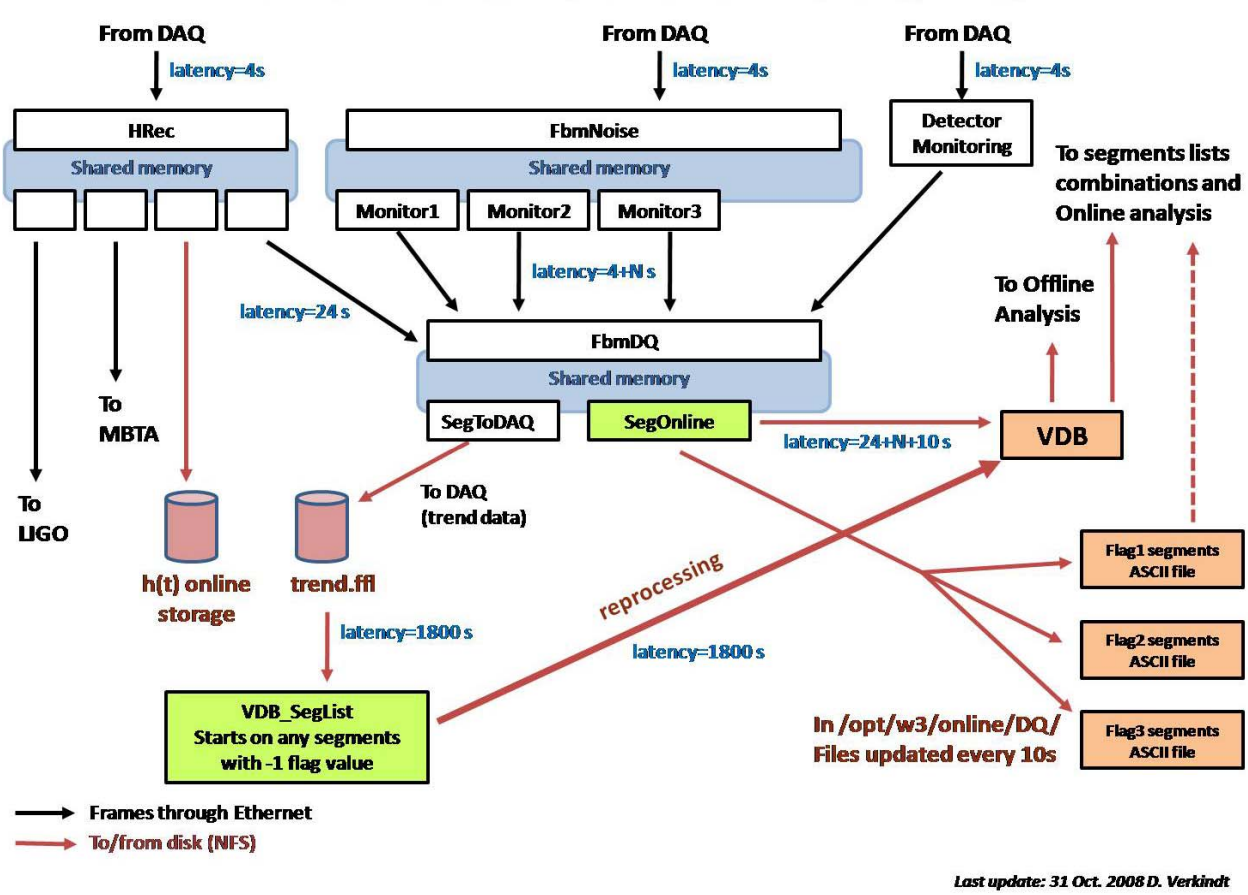

Fig. 3: Virgo online DQ generation lay-out for monitor taking data from DAQ

6.2 Sources of online DQ flags

The 3 main sources of online DQ flags will be the following:

1. **Online h(t):**
   h(t) is produced on a single machine (olserver9) using the DAQ tools (FdIO). h(t) quality flags are sent to the trend data. A modification of the Hrec process or an additional process to produce online the h-dependent DQs in an ASCII file of /opt/w3/DataAnalysis/online/dq can be envisaged.
   - input: frames from a shared memory
   - output 1: h(t) frame files written on disk
   - output 2: DQ related to h(t) quality stored in h(t) frames and sent to FbmDQ
   - output 3: h(t) and DQ sent to online analysis pipelines like MBTA
   - output 4: h(t) sent to LIGO
2. **Online detector monitoring:**
   Detector monitoring results are produced on one or two machines “a la olserver12” using DAQ tools (FdIO). Some of the flags listed in section 3 will be produced by detector monitoring and sent down to FbmDQ and SegOnline.
   - input: frames from a shared memory
   - output 1: all flags sent to DAQ and stored in trend data
   - output 2: some flags used for online DQ and thus sent to FbmDQ

3. **Online DQ production:**

We may also need to develop new monitors to generate the flags listed in section 3. We need to make an inventory of what is already produced by the existing online monitors. Most of them are supposed to use the FdIO library to get data online and their I/O will follow the implementation described in Fig. 3.

- input: frames from a shared memory (4s latency)
- output: DQ sent to FbmDQ

4. **"In-time" DQ production:**
   Some monitors will read data in-time from raw data files (latency=4mn). As shown on Fig. 4, they will output DQ segments in ASCII files, in /opt/w3/online/dq or directly feed VDB.
   - input: frames from raw data files (4mn latency)
   - output: DQ sent to ASCII files and copied into VDB.

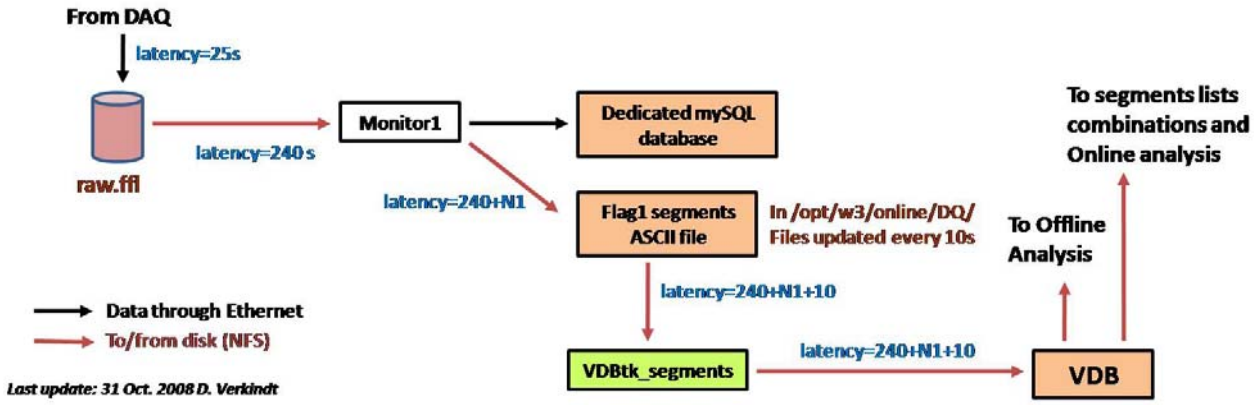

Fig. 4: Virgo online DQ generation lay-out. For monitors reading raw data files

6.3 Reprocessing procedure

If a process crashes or if the input data are missing, a reprocessing will be needed to have always in VDB the most complete DQ segments. The latency constraint on such reprocessing is not huge: a one day latency seems acceptable.

As shown on Fig. 2 it is proposed to periodically (daily) check in VDB for segments with value -1 (meaning no data or no information available) and to reprocess them in two steps:

1. read the trend data with the VDB_SegList process in order to replace the -1 values by 0 or +1. The mechanism to update VDB by reading the trend data already exists and is used to produce ITF_SCIENCEMODE_ONLINE, ITF_LOCK_ONLINE and INJECTIONS_ONLINE.
2. If step 1 has failed, it means that the DQ flag values are not in the trend data. This means that the monitor was not active (process down) or that the input channel was missing. In this case, an offline version of the monitor must be run on the raw data to produce the DQ flag values and an offline version of SegOnline must be run to rebuild the DQ segment.

If step 2 fails, it means that raw data are missing for the segment time epoch. No reprocessing can be done. But, there is a special case to consider: when raw data files storage on disk fails, the second data stream line allows to recover the missing raw frame files, sometimes a few hours (up to days) after the failure. We need to have a procedure in order to generate the

missing DQ automatically when it happens. When the raw data are missing, we should first check if corresponding trend data are available. If yes, we can generate missing DQ with them, otherwise, me must reprocess the missing DQ using the offline version of the monitors and using the rawback frame files as input. A specific process to manage such reprocessing needs to be developed.

It is assumed that all segments in VDB are continuous, which is the convention to fill VDB with segments.
[MAB: what about the case where we have non continuous segments, sort of holes. In that case the test of -1 would not detect a problem. Do we have a guaranty that SegOnline always generates “continuous” segments of +1, 0 or -1 values? If yes there is no problem. But what is foreseen for instance if SegOnline crashed or misbehaved for a while.]

6.4 online DQ storage in VDB

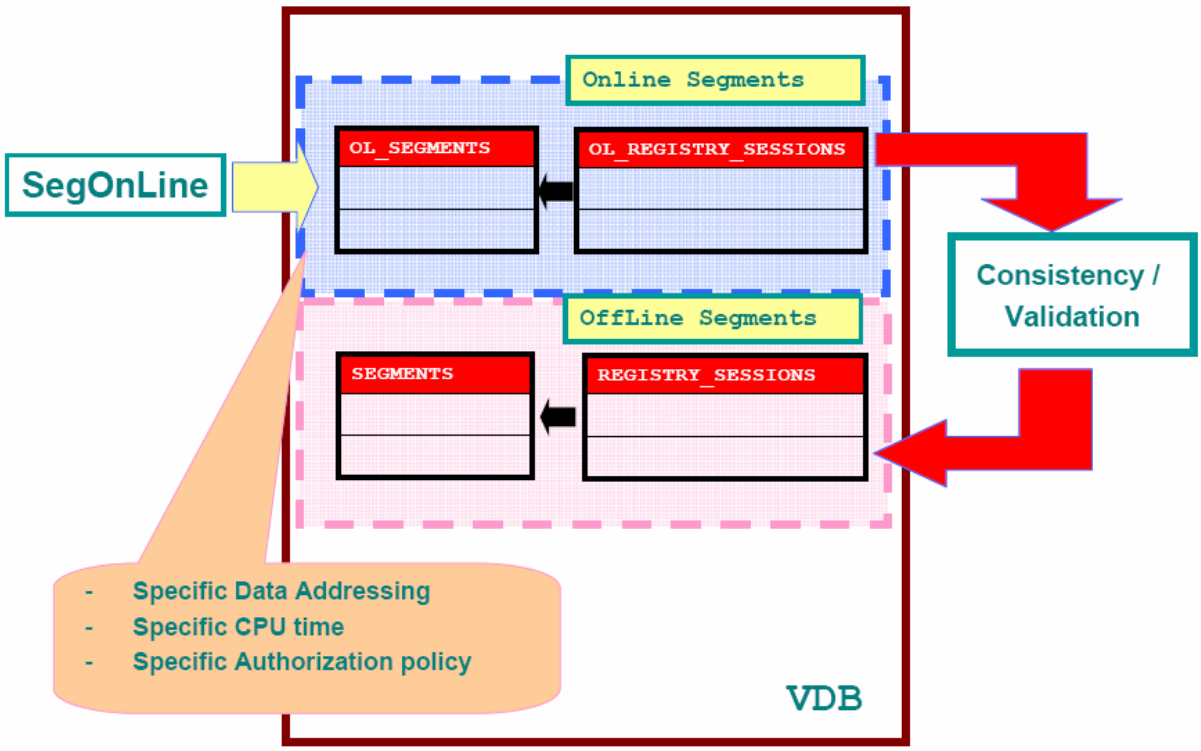

Fig. 5: VDB internal tables description to manage online DQ generation

For all DQ segment created online a replica of the “offline” existing tables will be created in VDB. Fig. 5 describes roughly the database table definition where the blue area represents the online VDB section and the pink area the offline VDB section.

This subdivision allows to customize and to control many parameters of the mysql server:

- mySQL is multi thread, thus a CPU can be allocated to online tables management.
- Query/memory optimization
- The authorization to this area can be independently tuned at the level of /host/user/connection
- Dividing logically the "Stable/Unstable" DQ tables from the "online" tables.

In Fig. 6, the data coming from SegOnline are stored in the VDB online tables.

The process described in section 6.3 checking daily the presence of -1 segments will access these "online" segments. Moreover an external process will verify the consistency between trend data files and VDB).

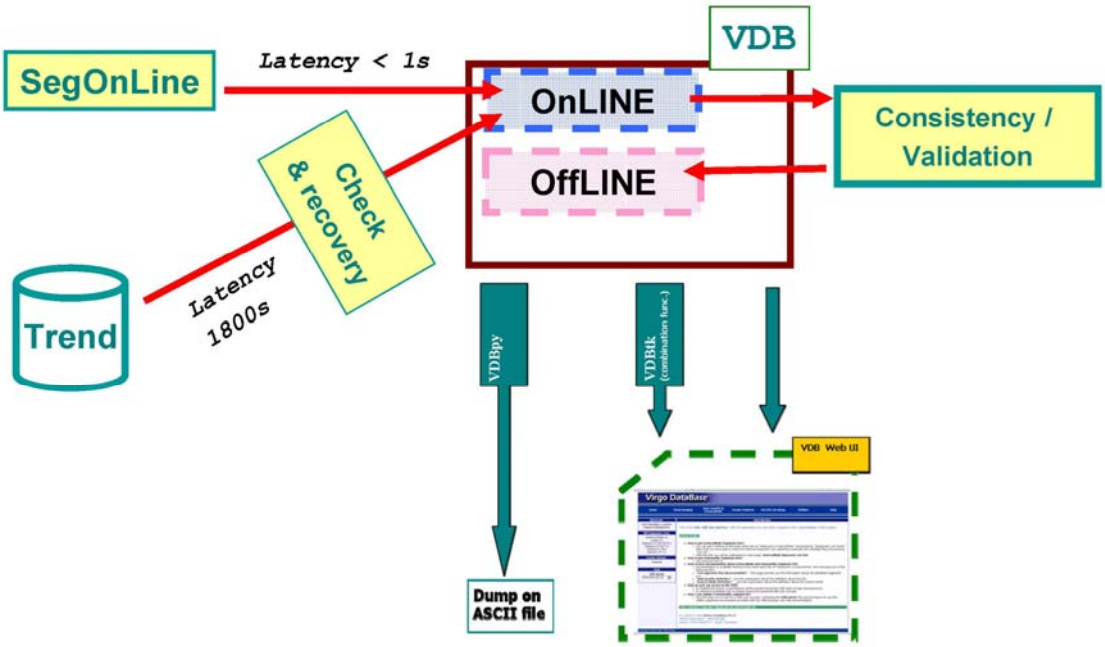

Fig. 6: VDB Inputs/Outputs for online DQ generation

Moreover all the VDB user tools are able to access OnLine/OffLine tables transparently, distinguishing the two types by the version tag (this is already done for the INJECTION, SCIENCE and LOCKING online segments lists).

An online segment list is regularly updated (addition of new segments), but can be as well corrected if a problem is discovered. The VDB API is being modified in order to be able to modify only partly the list and not being obliged to do a full upload.
In order to keep track of the correction made to an online list, a snapshot of the online list will be created each time a fixing is done. The online list (after correction) keeps the same version tag ("online"), but the old lists will be kept and accessible in VDB.

[MAB: what about the synchronization between the regular update (each 10s) and a fixing?]

On a monthly (?) basis "online" segments will be validated and offline replica tables will be updated. Each updated table is associated to a new version number, regardless the status stable/unstalble of the offline list.

Convention:

- Keep VSR2_Vx for stable offline version
- Use Vx for unstable offline list version
- "online" unique version for the online list (regularly updated)
- "online_<gps>" version for the list that is kept before the online list is fixed for a bug

7. Veto list production

In order to simplify the process and assure quality, one fast glitch finder algorithm, KleineWelle (KW) will be used to generate triggers for all ITF channels, online, at each ITF site at illustrated in Figs. 1 and 2. For Virgo, about 200 channels will be surveyed. The vetoes will then be studied with common algorithms ranking channels according to their usefulness as already developed and tested during S5/VSR1. Different strategies/algorithms will be applied for burst and inspiral searches.
A small latency (below 5 minutes) is mandatory for the KW triggers production. A latency of a few hours up to 1 day to generate veto segments is accepted.

Furthermore, the KW triggers will also be used to generate statistical information on the glitch rate for each channel in order to give monitoring information in the control room of any change that could be coupled with an improvement or deterioration of the dark fringe noise. A low latency (~1 hour) could be useful for commissioners to see if a fixing improves or not the glitch rate. Tools (statistical information displayed on ascii files) have been already developed for S5 [4]. One should envisage to adapt these tools to Virgo, or to develop specific tools (may depend on available manpower).

7.1 Implementation scheme

It is planned to get input data directly from DAQ, thus to interface KW with the FdIO library. This means to develop an interface between FdIO and DMT which is the LIGO tool in which KW is embedded. The main path is to include DMT in the Virgo software management environment (CMT) to use it like an external library and then to create a Virgo version for KW using FdIO instead of DMT to access data. All the needed channels will be sent from FbmNoise or from others specific providers down to a specific machine managing the cluster of machines where will run the KW processes.

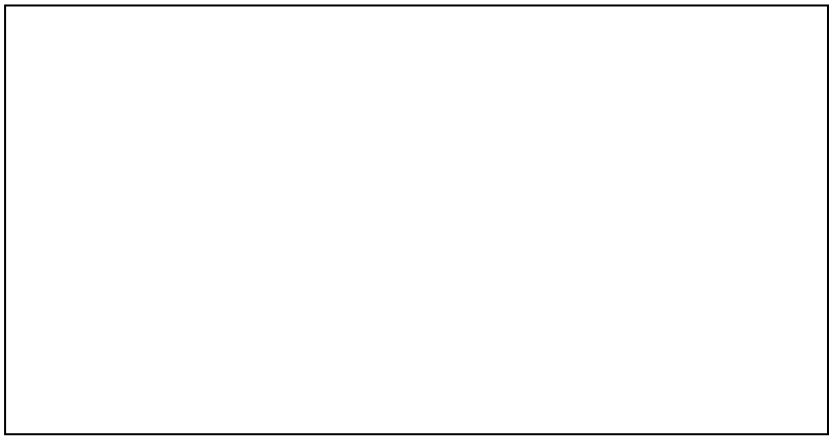

Fig. 7: Implementation scheme of the VSR2 online vetoes production

7.2 KW triggers production

The KW triggers will be generated using the list of channels described in section 4. This list may evolve during the first weeks of VSR2. For each channel, a minimal threshold on the KW significance of 50 will be applied. The frequency range will be tuned from 10-50 Hz up to a maximum of 2kHz (or $f_{sampling}/2$ if below). Some channels could used a more restrictive sets of parameters like magnetic probes which could be limited to 600 Hz to avoid known glitches at high frequencies.

The format of the KW trigger file is still under discussion among LSC-Virgo, the two main possibilities are ASCII file (one per channel) or XML (one file per channel and/or merge into a single file for all channels).

7.3 Veto segments generation based on KW triggers

Given the manpower limitation it's wise to consider one algorithm per type of searches, burst or inspiral. It will be wise to have a LSC-Virgo common veto development strategy as for VSR1 in order to simplify the implementation in the analyses. But any new idea or development is welcome.
As for DQ segments, vetoes based on KW triggers will be stored in VDB.

[MAB: we need to identify for each search, someone who is willing to take care of the implementation and monitoring daily during the run the output of these algorithms.]

7.4 Other glitch finder algorithms

7.4.1 WDF (Elena, Marina) [5]

7.4.2 OutlierMoni (Didier) [3]: this algorithm does a whitening of the data, using a moving averaged spectrum over about 20s. The result can be compared to a threshold to generate triggers to be used as vetos. The outlierMoni output can also be used (as it was in VSR1) to provide a monitoring of the dark fringe stationnarity (http://wwwcascina.virgo.infn.it/MonitoringWeb/Noise/outlierMoni/archive/VSR1).
It provides no frequency information of the triggers but can be used to have a first estimate of the glitchiness of the dark fringe. But VirgoHACR or Qonline would certainly provide better information. Nevertheless, outlierMoni is a low CPU consumer. It can be used ot provide a first estimate of the withened dark fringe and to provide some statistics about the noise stationarity. outlierMoni can also be used for online vetoes if any complementarity with KW is found.

7.4.3 VirgoHACR (Gabriele) [3]

7.4.4 Qonline (Shourov + XX) [9]

[MAB: What are the plans how to use the output of these algos?
To be completed by the people taking care of the algos.]

8. Data Quality checks in the control room

In addition to the production of DQ or vetoes useful to data analysis, we should define a set of information to be provided in control room that can help people in shift to analyse in-time the ITF behavior and the events detected by the online analysis. Web pages, plots and statistics, periodically updated, will be developed. A screen in control room will be dedicated to each of the following summary web pages.

8.1 DQ monitoring web page

A single web page will show a summary of DQ online, with the following information:

- Status (value, latency, deadtime) for all the "basic" segments and online DQ flags with links to the segments list over the last 24h.
- Status of each monitor that produces a DQ flag.

This is illustrated by Fig.7 which gives an example of what could be such a page.
In addition, a web page will show plots and statistics (per day / per hour) of the effect of the DQ on the generated online GW burst and inspiral triggers.

Fig. 7: example of online DQ summary web page

8.2 KW monitoring web page:

Finally, we need also to develop web pages that gives statistical information derived from KW triggers and information about glitch rate in the auxiliary channels that can play role in the quality of the dark fringe. We should follow what LIGO has developed for that! (see http://lhocds.ligo-wa.caltech.edu/scirun/S5/SciMonCheck/KleineWelleGuide.html).

8.3 Noise web page:

In addition, a noise monitoring summary page will provide some detailed information that can be correlated to DQ:

- glitchiness (HACR), stationarity (outlierMoni) of the dark fringe, …
- time-frequency plots of dark fringe signal for the last event seen by an online analysis, with link to Qscan of the event
- table of the 10 last events seen by the online analysis with information like:
    - Horizon or SNR
    - Event is vetoed by a DQ segment or not
    - Event is vetoed or not vetoed by a KW veto (or any other veto)
- other proposals (see for instance [5] from Elena). The present list is far from being complete especially concerning statistical information that indicates that the Virgo noise is degrading.

Fig.8 gives an example of what could be such a page. An alternative example is provided by Elena at http://wwwcascina.virgo.infn.it/DataAnalysis/Noise/NMW/html/index.html. What will be shown and which tools will be used is under discussion and certainly require an input from commissioning and data analysis experts.

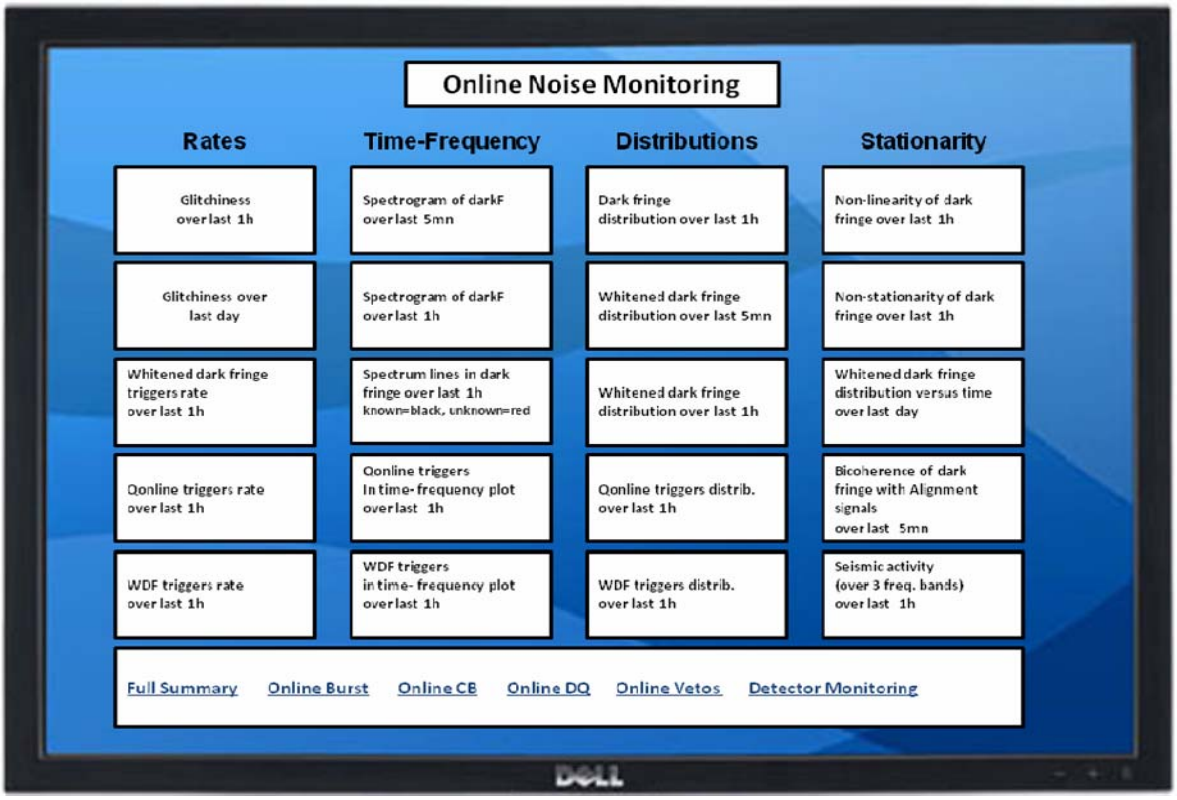

Fig. 8: example of online noise monitoring summary web page

9. Monitors configurations automatic archive

We need to record all changes of parameters (threshold, frequency bandwidth, …) of the DQ monitors. Some information is now recorded in the logfile of the processes but the information is buried in lots of log output and logfiles are deleted after a while.

A proposed solution is, using CVS, to commit changes of the configuration files in /virgoData area each time the interferometer is relocked and to be allowed to change configuration files only when interferometer is not locked. We need a tool for this and a volunteer to develop and maintain it.
[Didier: discussion with Franco Carbognani may help in choosing or developing such a tool. Also, A. Masserot said that the use of Cfg provides automatically the needed snapshot of the configuration each time a monitor is started.]
[MAB: how would that work?]

10. Tasks and organization

To organize the real work to do, we have defined a list of tasks and a list of people who agreed to take responsibilities on at least one item (to be completed, any volunteer welcome!). Part of this list should be cross-checked with diagram of Fig. 2 which is periodically updated.

DQ online production: Didier, Leone, Frederique, Mab, xx?
- Architecture, connection to DAQ: Didier
- Development of monitors producing online DQ: see the list in section 3
- Development of SegOnline that feeds VDB online: Didier, Leone
- VDB online/offline management for feeding and users access: Leone
- Daily check of the -1 segments in VDB and launch+management of reprocessing (maybe just a bash script and the use of VDBtk_segments): Didier, Leone
- DQ reprocessing from trend data (SegList): Didier, Leone
- DQ reprocessing from raw data or from rawbackup data if raw data are missing (DQ Monitors + SegOnline in reprocessing mode): Didier, Leone, xx?
- A process to read the segments in ASCII files produced by "monitors not using FdIO" and to feed VDB with them (VDBtk_segments): Leone
- DQ segments list for online analyses: Mab, Frederique
- Exchange of DQ with LIGO: Leone, xx?
- Exchange of h(t) with LIGO: Shourov, Benoit, Didier

Vetoes online production: Nicolas, Benoit, Elena & Marina, Didier, xx?
- KW triggers generation: Nicolas, Benoit, xx?
- WDF triggers generation: Elena, Marina
- outlierMoni triggers generation: Didier
- Q online trigger generation: Shourov + xx?
- Veto segments generation (implementation of the veto algorithm for burst and CBC): xx?
- Feed of VDB, exchange of vetoes with LIGO: xx?

Data quality web pages in control room: Didier, Elena, Mab?, Nicolas?, xx?
- DQ web page: Didier, xx ?
- Noise web page: Elena, Didier, xx?
- KW web pages: Mab?, Nicolas?, xx?

Monitor configuration files CVS automatic archive: xx?

11. References

[1] The LSC-Virgo burst group, “Real-time search for gravitational wave transients during S6/VSR2”, August 6 2008.

[2] The LSC-Virgo inspiral group,
http://www.lsc-group.phys.uwm.edu/daswg/wiki/S6OnlineGroup

[3] Didier Verkindt, “Online Monitoring: status after VSR1 and plans for VSR2", version 2, Sep 11 2008.

[4] KW triggers web pages at Livingston and Hanford: http://lhocds.ligo-wa.caltech.edu/scirun/S5/SciMonCheck/KleineWelleGuide.html

[5] Elena Cuoco, “Virgo online Noise monitoring for VSR2/S6”, version 1, Aug 28 2008.

[6] The Frame Distribution Library (Fd, FdIO), Virgo note VIR-087A-08

[7] Frames Shared Memory Library (FdShm), Virgo note VIR-086A-08

[8] Virgo Data Base reference manual, Virgo note VIR-XXX-XX