Status and (near?) Future Plans for Data Quality Monitoring

TRG/DAQ Workshop

29.11-2.12.2022, Nara

Björn Spruck

Content

- Introduction / Recap
- Results of BPAC and internal discussions
- Technical improvements
- Short term plans

Data Quality Monitoring

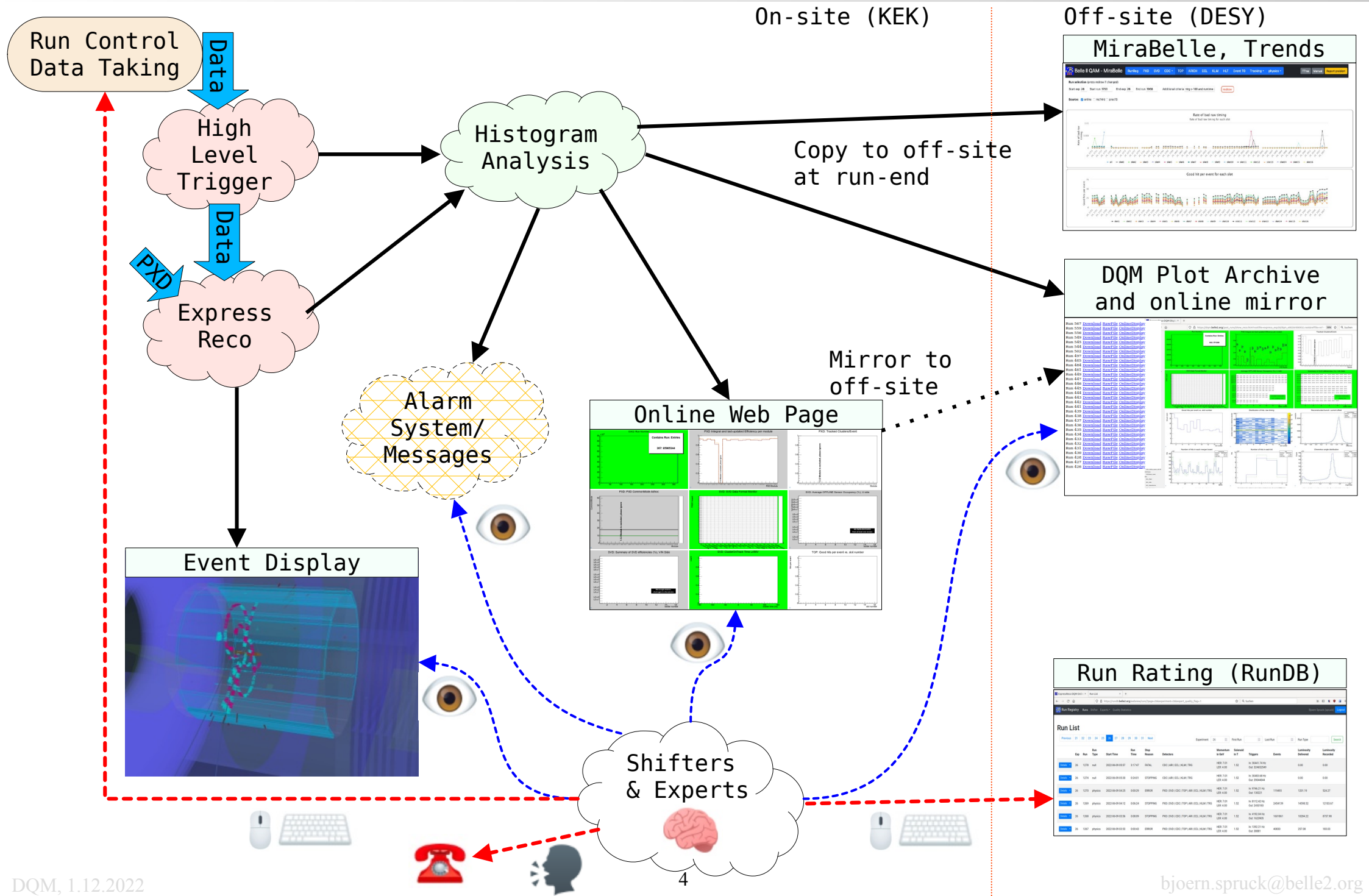
"Online"

- Available during the ongoing run
- Shifter can intervene
  - → stop the run
  - → call expert
  - → restart when fixed
- Prevent that we take bad quality data!
- Fast developing/severe issues e.g. from SEU

"Offline"

- Available only after run has stopped
- → data is already on tape
- Archive
- Select good/bad runs
- Post-mortem analysis of problems
- Slowly developing issues (trends)
  - Minimal impact on data quality

Recap - Schematic Overview



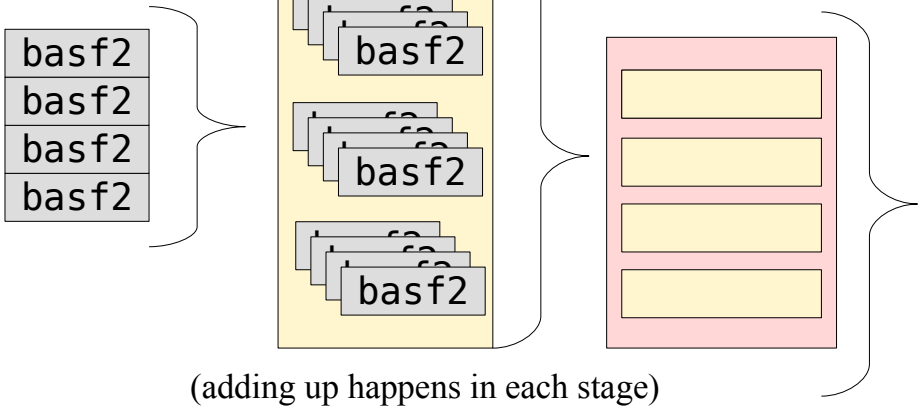
Technicalities: Histogram Flow

Filling and adding on HLT / ExpressReco

Histograms filled in each basf2 analysis. Several threads per worker node

Several workers in HLT/ERECO unit

Several HLT/ERECO units

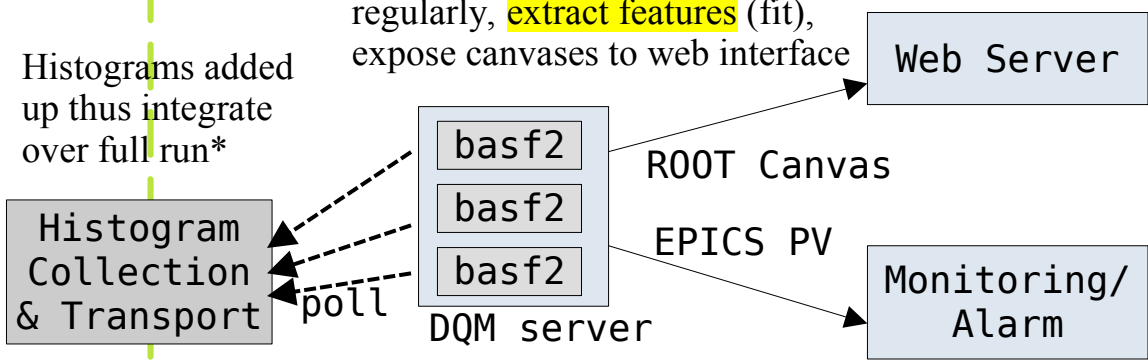


(adding up happens in each stage)

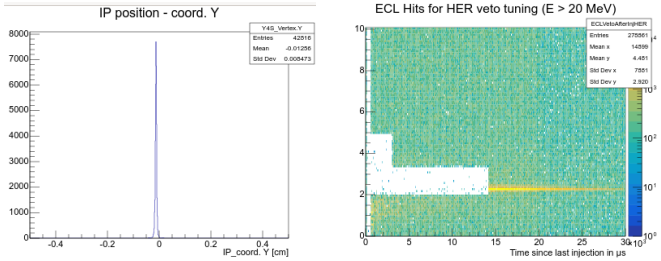
Histograms added up thus integrate over full run*

Analysis on DQM server(s)

Histograms read and processed regularly, extract features (fit), expose canvases to web interface

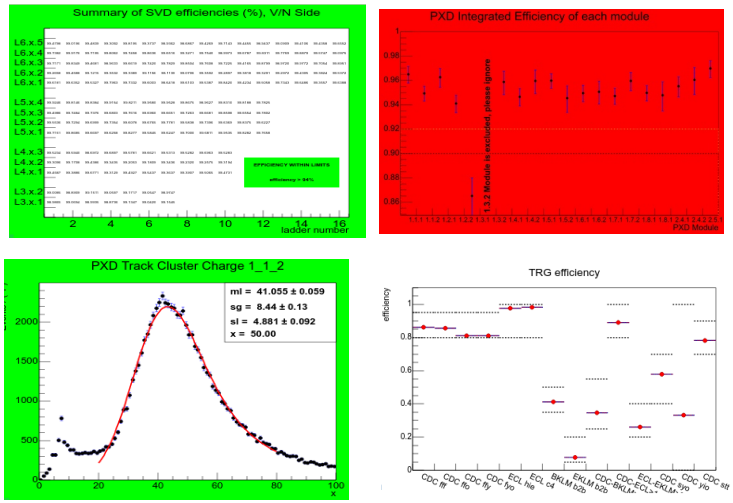


Only plain 1d and 2d histograms can be added up



Limits apply for overall histogram size as regularly adding up consumes **CPU** and **network** resources

```
>3800 histograms on HLT, 41 MB (6.5 MB in file)
>7900 histograms on ERECO, 88 MB (16 MB in file)
```

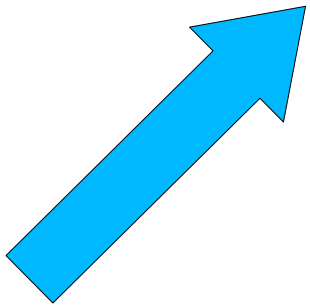



```
O(hundreds) histograms used in analysis
```

* Run: One run may last O(min) to ~~8h~~ → 4h

Technicalities: Histogram Flow

To check:
It seem we do not exactly know what our current limit is, neither what effect (in term of CPU/network performance) we currently have on HLT (after change to 0mq)
Effect of changes on ERECO, adding of nodes/units?
→ automatic test/measure in basf2 CI/CD?



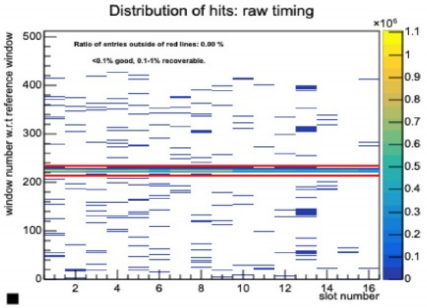
Limits apply for overall histogram size as regularly adding up consumes **CPU** and **network** resources

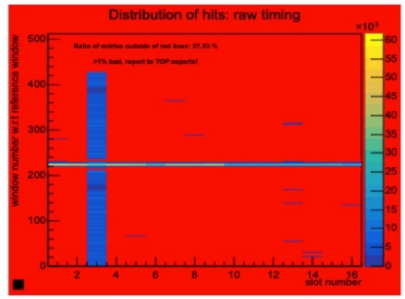
```
>3800 histograms on HLT, 41 MB (6.5 MB in file)
>7900 histograms on ERECO, 88 MB (16 MB in file)
```

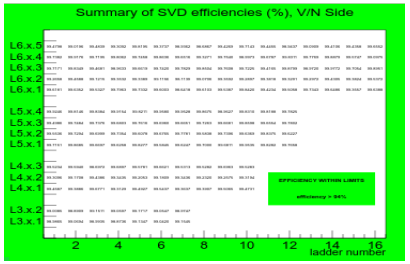
```
O(hundreds) histograms used in analysis
```

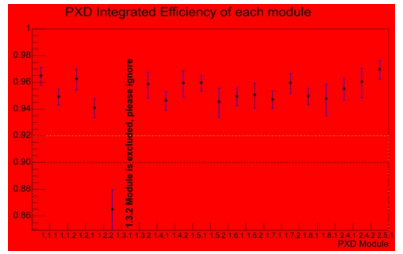
Histogram Analysis and Presentation to Shifter

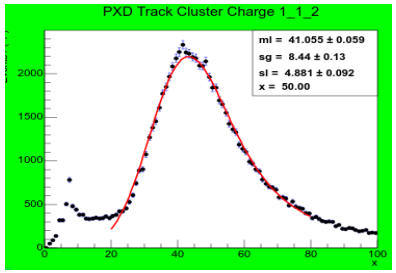
- Plotting
  - Overlay with **reference plot** (default for 1d-plots if no other analysis)
  - Labels, High-Low margins
- Analysis
  - Extract mean/width
  - Fitting
  - Efficiency (dividing histograms)
- Consolidate (merge result from different histograms into a new plot)
- **Automatic detection of anomalies!**
  - Colorize histogram to visual alert shifters
- Export observables and status ("color") to EPICS (for alarm and monitoring system)
  - → ring alarm for expert, post message to RocketChat for shifter
  - Continuous trend (e.g. IP position for SKB)

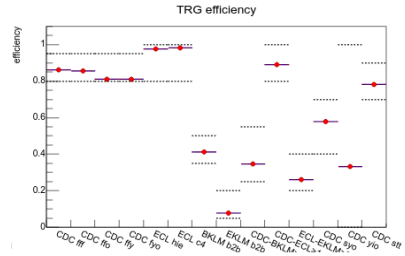


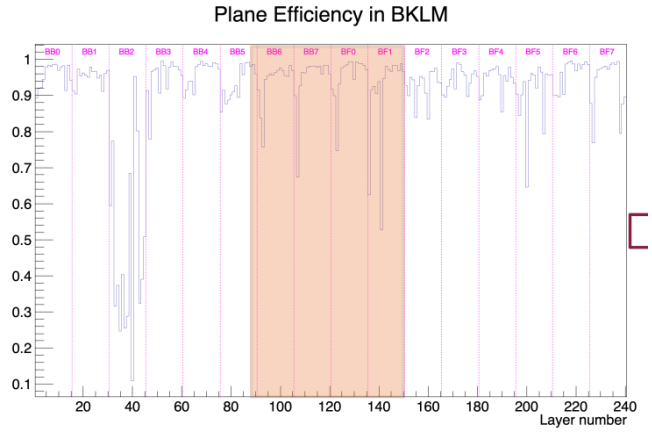












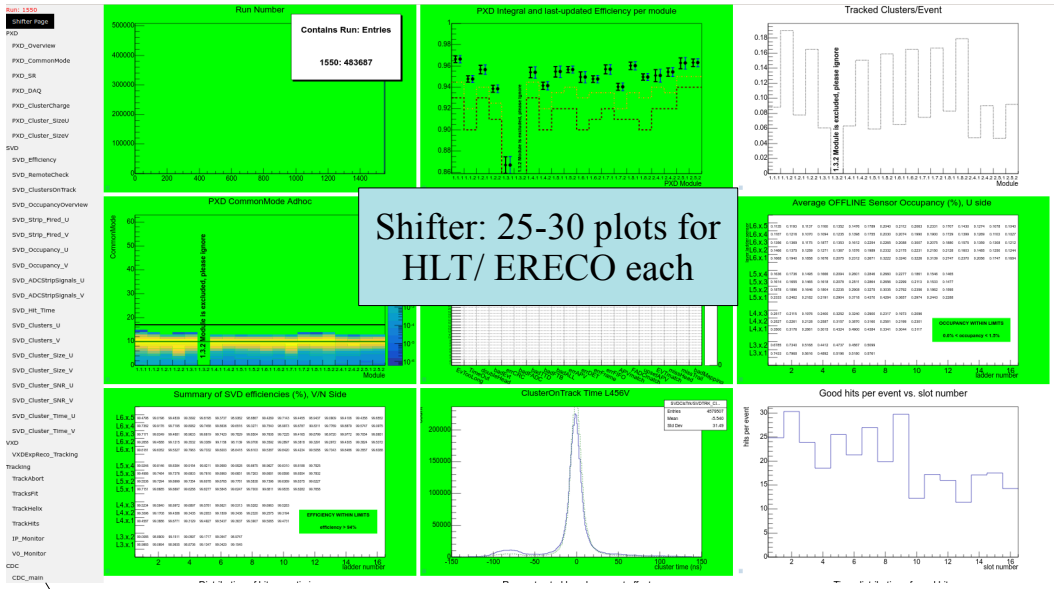
Web Server and Interface

- Separate servers for HLT and ERECO
  - Shifter and expert pages
- Plots/location/layout by JSON templates
  - Each plot need to be added manually

- Now: using root embedded web server
  - Canvases send to root process and served directly from memory
  - Original idea: make use of jsroot and its interactivity (zoom, fit, etc) ← not used!
    - Interactivity unusable due to auto-reload of web page
  - Performance basically o.k. but issues (get slower over time?)

- Update plan: move to a dedicated web server and provide static images
  - → Better server performance
  - → optimize load and traffic by providing only updated plots

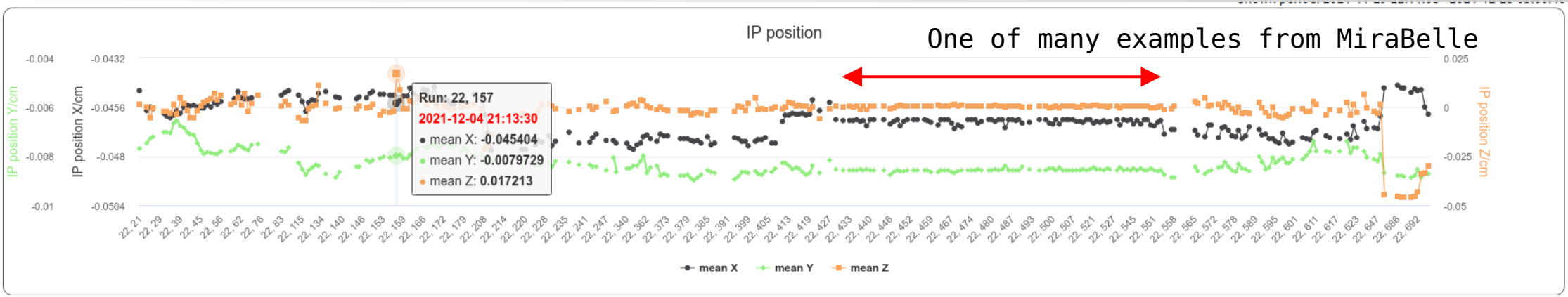
Decouple analysis from web server.



- Use case: *slowly* developing trends

- Agreed to use MiraBelle as a <u>common</u> monitoring tool for all detectors in 2019
  - Web based, using external javascript packages (highchart, datatables, jsroot etc)
  - Selection of run ranges and selection criteria
  - Run based: observables extracted at the end of a run from DQM plots
    - Plots can be embedded (MonitoringObjects)
  - Integrates information from RunDB, beam condition etc, MC & prompt

DQM Quality Problem – Mixing Runs



- Exp 22 Run 433-549 have mostly unusable ERECO DQM plots (HLT plots were not affected)
  - Plots from earlier run (431?) with large statistics are added into ongoing run
  - **Visible in unrealistic statistics** for short runs and lack of variation (but neither shifter no expert noticed!!!)
  - → 4 days running blind (mainly for PXD, as PXD has no HLT plots)
- Most probable reason: second ERECO unit crashed but still provided (old) plots
- The actual problem of mixing old plots into new run has been observed several times before (HLT and ERECO), often seen when a minute long run has unrealistic statistics.
  37th B2GM 2020_11_10_dqm_pxd.pdf
  - Until now, no proper way to **detect** it neither online nor offline. (Update: There is a automated plot analysis since 2022)
  - Corrupted DQM files/plots not cleaned up in archive → re-checking runs later is based on wrong assumptions
- Some histograms had run number embedded into histogram titles already. If number differ from ongoing run, this was a clear indication. But: which title survives the histogram merging stage depends on the order, thus not really useful

Insufficient (internal) monitoring, reliability

We were lucky that only ERECO was affected. This resulted mainly in PXD (and physics) not watched for three days by the shifter. Could have been much worse.

Shifters were not aware of what they were looking at. A 4h statistics after 5 minutes of run? Statistics of ERECO plot >> Hlt Plot?
Can we RELY on CR shifters to detect issues?

List of Issues and (Missing) Improvements

Operation

- Analysis gets slower over time → workaround restart every n hours (but better to find & fix the actual reason)
- Several minutes delay between run start and histograms appearing in ERECO (=empty plots for short runs) → **update of ERECO and storage framework**
- **Low statistics in physics channels → skims**
- Web server gets slower over time → use proper webserver and updated only changed content
- Histogram bleeding into next run → detection and msg to alarm shifter, final fix by ERECO update
- Proper **test bench** ← open
- Automatic copy of histograms to off-site (DESY, kek) cumbersome, sometime stuck. Several script running (e.g. for MiraBelle) ← improvements
- **Reduce manual interventions**
  - In normal operation
  - Offline site and MiraBelle on experiment number change
- Version control of scripts on dqmsrv
  - Strategy for server failure

Detector side

- **Review** what is shown and if important things are missing
- **Employ** our **existing tool**-set
  - Use **automatic anomaly detection** where possible to help shifter notice issues ← limits, colorize, alarms
  - Export meaningful observables to MiraBelle, document

Framework

- Run integrated → Time slice delta histograms analysis
- Lower barrier for new developers → consolidate, simplify, cleanup, good examples
- Provide common functionality within the framework to save work on detector developer side
  - Shift man-power to improve on the framework itself!
  - Goal: Hist analysis without writing (too much) code
- Use configuration file for setting, limit and EPICS (high level abstraction)

Shifter & GUI

- Rethink shifter workflow and presentation
- Consolidate web-pages (plots at the top get more attention)
- Automate to avoid subjectiveness, raise attention, and help the shifter decide
- Trend plots: MiraBelle (& CSS?)
- → Discussion needed

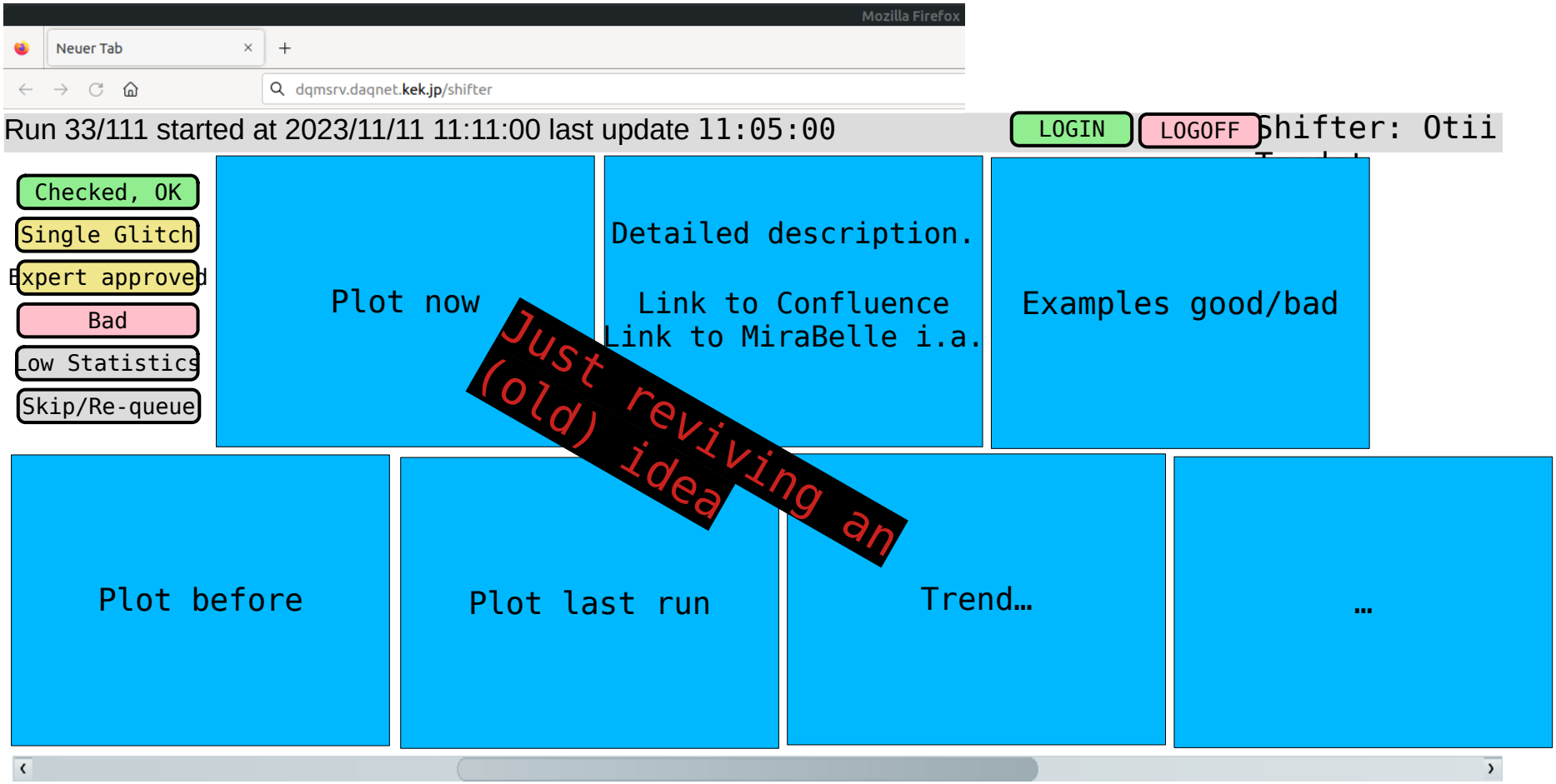
Suggestions from BPAC and other Discussions

- Integration into alarm system
  - → better automatic anomaly detection
  - Faster reaction (+ automatic actions? tbd)
  - Reliability → independent of shifter experience/decision
- Review **what** we have to monitor per sub-detector.
  - Do we have all relevant observables?
  - Are they monitored in a suitable way and understandable by shifter?
- Improve dqm-shifter interface
- Importance of (long term) trend plot monitoring (MiraBelle)

- From/on my list:
  - Simplify development (for adding new plots, analysis, → alarms)
  - Bug-fixes and code clean-up ☑
  - Time slice "delta" analysis (support in base class) and update tagging ☑
  - Improve operation

All-in-One Web-Gui

- Enforce that all plots are checked and signed-off (like in RunDB)
- Collect all plots in a central page (HLT, ERECO, history, trend, ref, …, description)
- Just some idea, need reiteration



Clean-up, Bug-fixes, Documentation

- Clean-up was triggered by debugging some strange occurances of mis-colored canvases of PXD DQM plots (found: one analysis module going rampart and modifying canvases not belonging to itself. Same bug has been copy-n-pasted to other modules.). Poor code quality, QC and missing tests are the reasons.
  - Bug was fixed at that time and a bit of cleanup done. But loose ends remain.
  - I am sorry it took me 9 month to finish the clean-up.
- Now, all “findHist” or”getHisto” function have been moved to the base class. Please only use provided functions and do not write your own code.
  - Unnecessary and perhaps even dangerous code has been removed (searching full memory for ref histograms)
  - Function now only do what their name implies. (maybe “find” prefix may not be proper and is better changed to “get”)?
- More work
  - Start to clean up warn/info/error messages (“2GB log files”)
  - Documentation

```
// Functions:
hist=findHist(...);
hist=findHistinFile(...); // for reference plots
canvas=findCanvas(...);
hist=findHistinCanvas(...);
// new:
hist = getDelta(...);
```

Tagging Updated Histograms/Canvases

Big Update (1)

Canvas update tagging

- Issue:
  - Currently, analysis modules are run, independent of if the histogram was actually updated.
    - Analysis, fitting etc, are done and canvas is updated even so no update is necessary
    - → **unnecessary** CPU **load** on the analysis server
  - Currently, **all** canvases in memory are send to web-server, independent of if they have been updated or not (= if the analysis module has not updated the canvas)
    - → **unnecessary network traffic** from analysis chain to web-server
    - → **unnecessary network traffic** from web-server to clients
    - → **load** on the web-server
- Solution:
  - Keep track of histogram content (entries) in base class and publish an interface to analysis side for requesting only updated histograms
  - Keep track of canvas status: analysis modules need to tag canvases as (not) updated. Only updated canvases are transferred/exported further. The web-server is able to save resources by not re-sending unchanged content to clients.
  - (even more if we change to a static web server as planned)
- Compatibility: In the default setting, no change to current behavior
  - User code should call a function to explicitly state if canvas has been updated or not.
  - Output code checks the flag and behaves accordingly

Small changes to user code needed:
(only schematic to show the concept)

```
// get only updated histograms
auto hist=findHist(histdir,histname,true);
if(hist==nullptr){
    UpdateCanvas(m_canvas->getName(), false);
}else{
    // update canvas m_canvas e.g.
    m_canvas→cd();
    Hist->Draw("hist");
    UpdateCanvas(m_canvas->getName(), true);
}
```

Delta Histogramming

https://agira.desy.de/browse/BII-9546

Big Update (2)

- Mitigate the integration effect on histograms in long runs.
  - Used in IP position monitoring and PXD for years, but common interface was missing in base class until now.
  - **No change in default behavior**, need to be enabled by user and explicit change to getDelta() instead of findHist().
  - Flexible configuration per histogram (algorithm, number of entries or events, etc). Examples are provided.
  - Parameters set in initialize (for now) using parameters given in python script. Todo: Better from configuration file (plan for near future)

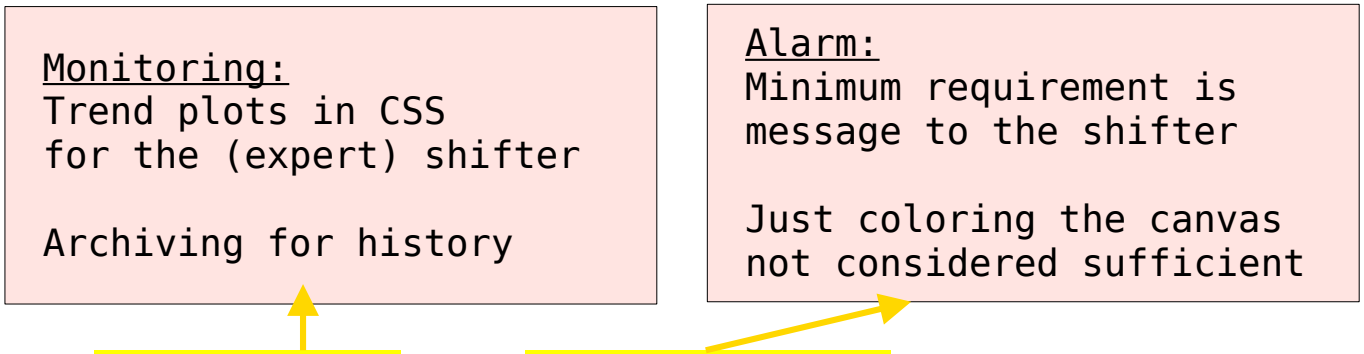
```
void DQMHistAnalysisIPModule::initialize()
{
  B2DEBUG(20, "DQMHistAnalysisIP: initialized.");

  m_monObj = getMonitoringObject("ip");

  addDeltaPar(m_histoDirectory, m_histoName, 1, m_minEntries,1); // register delta
```

```
void DQMHistAnalysisIPModule::event()
{
  auto delta = getDelta(m_histoDirectory,m_histoName);
  // do not care about initial filling handling. we wont show or update unless we reach the min req entries
  UpdateCanvas(m_c1->GetName(), delta != nullptr);
  if (delta != nullptr) {
      m_c1->Clear();
      m_c1->cd();// necessary!
      delta->Draw("hist");
```

Simplify EPICS Monitoring & Alarm System



- Simplification of EPICS usage as **monitoring** and **alarm system** interface:
  - DQM modules run as "client", PVs must be created in a server task elsewhere.
    - (currently this is done on a PXD host for PXD and IP, and other hosts for SVD and other detectors) → would be nice to have this central and easily adjustable.
    - Idea: **parsing configuration files** (JSON) instead of manually changing and restarting
      - → python ioc currently tested on pxddqm server
      - Contains EPICS PV names plus limits/alarms, …
- Same JSON file used for configuring, e.g. Delta histogramming, common fitting and testing ("simulated" analysis)

- (Optional) add EPICS support to a base class for easier handling

Reminder: Code-free Histogram Analysis

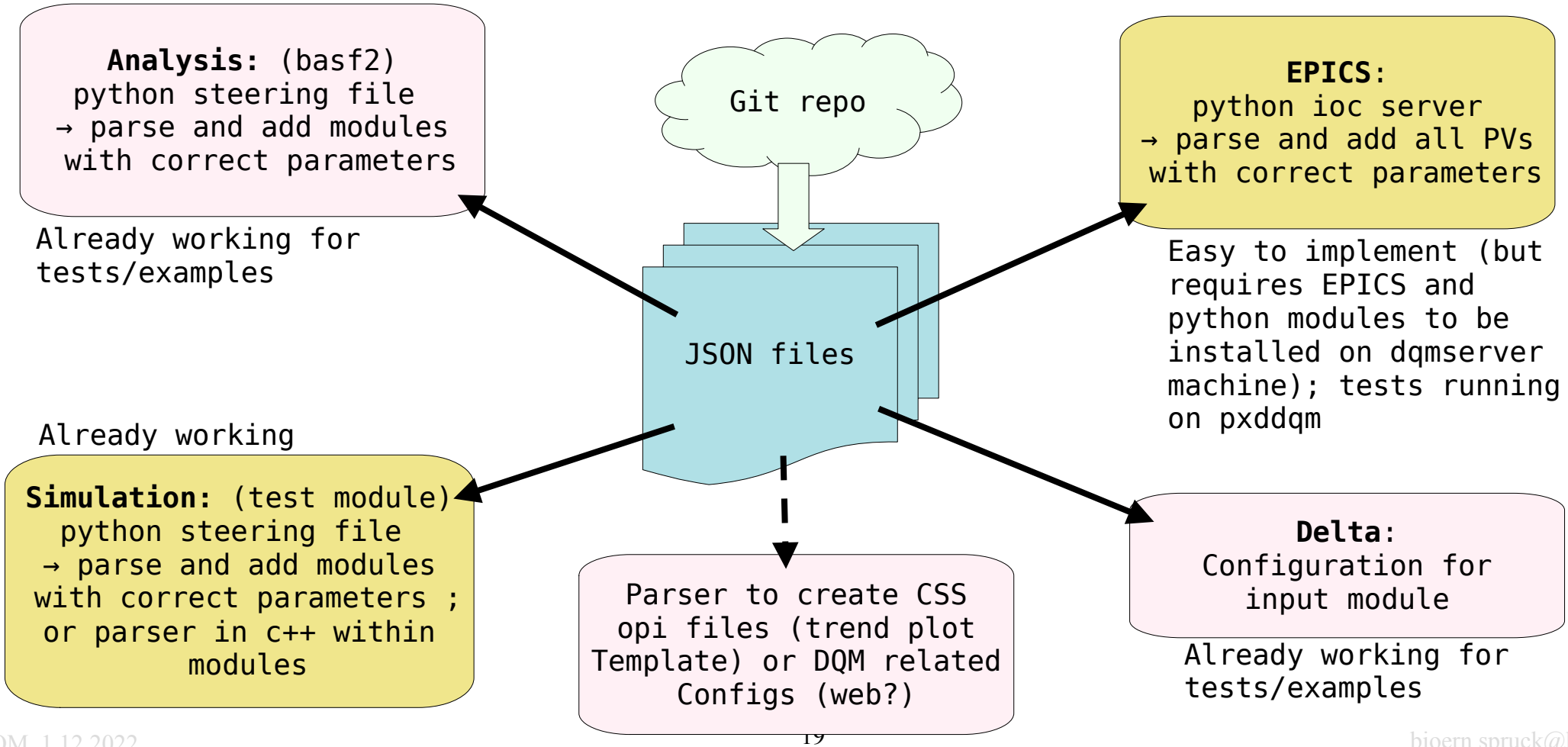
- Idea: Adding simple histogram analysis without writing any code.
  - Framework provides the functions/implementations, detectors use provided functionality
    - E.g. take mean/width/median/percentile of distribution, fit a function etc
  - Minimize hurdles for beginners as no code (C++) has to be written!
  - Either from python basf2 steering file (module parameters) or configuration file
- Technical details and complexity should be kept within the framework, detector liaisons should concentrate on what to monitor

- → Fast adaption to changes as no new code needs to be reviewed, verified, and rolled out.
  - Avoid release cycle, emergency release, etc in case you "just" need to change limits for histograms or add an existing one for shifter verification.

- Have a common set of modules "examples" which could perform the common analysis
  - We already had this in mind when writing the first example analysis modules in 2017
  - Was not taken up by developers, prefer to write their own complex code instead of re-using existing one? To complex or inflexible?
  - Revive and improve

Configuration by JSON Files

- Suggestion: Split python steering code and configuration
  - → it is safer to change configuration, less error prone than changing the steering script
  - Configuration files can be easily and automatically checked (structure and content → json schema) before commit (pre-hook) and before merge (gitlab) and deployed
  - Configuration files can made per-detector for easy maintenance.



Summary

- LS1 opportunity to improve DQM taken into account three years experience
  - Possible improvements in all levels: operation (technical), framework (bugfixes & features), GUI, …
    - Documentation and stream-lining code to make it easier to read/understand
  - Automation and alarms in case of anomalies a must. Reduces shifter dependency
    - Tools are already available
  - Consider our future style of monitoring/shifting.
    - Better adaption (integration) of DQM web interfaces to shifter workflow desirable (plots, references, examples, history)

- Detectors:
  - Evaluate available observables (per detector) for sensitivity and more added if important monitoring is still missing → **bring detectors on *same level* of monitoring**
  - Add “alarms” for shifter (beside coloring canvas)
  - Extend use of features: delta histograms, EPICS export
- Reduce hurdles for developers: complex, many manual steps involved for adding new plots
- ERECO update: ZeroMQ and use of physics-skims

Slide Intentionally Left Empty

Thank your for attention,
Comments and discussion is welcome!

Mailing List

- Mailing list confusing: “software-dqm-17gcr”
  - Neither it is 2017, nor related to global-cosmic-run nor is it “software” only
  - Suggested to either rename or create a new one e.g. only “dqm”
  - From DESY mailing list server both options are available, rename (and even keep an alias) would preserve list archive.
  - Better name? As it covers also software, operation and infrastructure, not sure about applicable prefix.

Missing: Test Benches

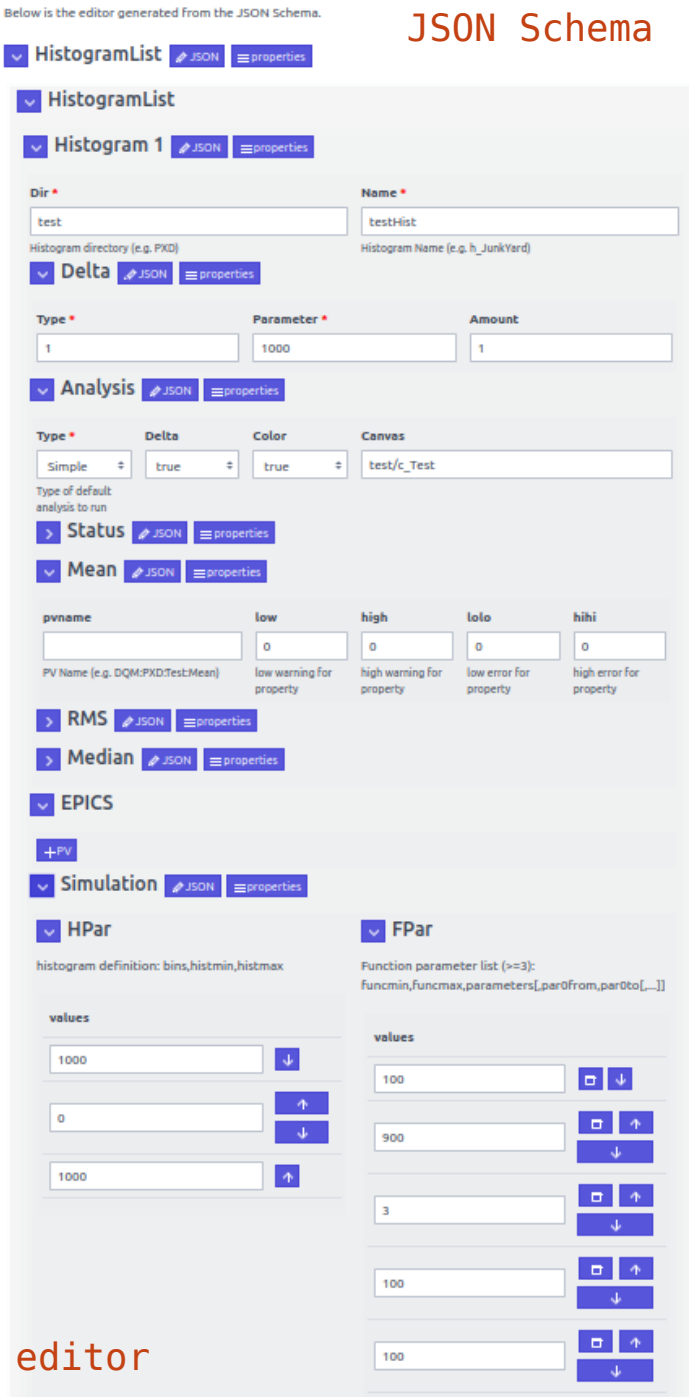
- Testing changes on DQM is not straight-forward
  - During LS1, we could use real dqmsrv for testing, including full chain of export to EPICS+alarms
  - Re-processing of data files from e26 on ERECO?
- How to have a full test bench environment in parallel to running and **not** interfering on 2024?
  - → spare/test HLT unit? gitlab container?

Why JSON & JSON Schema

- Using JSON files has more advantages:
  - JSON is easy editable text file (e.g. for copy`n`paste)
  - Automatic syntax check and against JSON schema
  - Web GUI builds automatically from JSON schema, including documentation and automatically checks input against schema

```
{
  "HistogramList": [
    {
      "Name": "testHist", "Dir": "test",
      "Delta": {"Type": 2, "Parameter": 500, "Amount": 10 },
      "Analysis": {"Type": "Simple", "Delta": true, "Color": true,
        "Mean": {"pvname": "Test:hist1:mean"},
        "RMS": {"pvname": "Test:hist1:rms"},
        "Median": {"pvname": "Test:hist1:median"}
      },
      "Simulation": {"HPar":[1000,0,1000], "Fill": 500, "Underflow": 100, "Function": "gaus(0)",
      "FPar":[100,900,3,100,100,400,600,30,70]},
      "EPICS": [
        {"pvname": "Test:hist1:mean", "low": 450, "high": 550, "lolo": 420, "hihi": 580},
        {"pvname": "Test:hist1:rms", "low": 45, "high": 0.55, "lolo": 40, "hihi": 60},
        {"pvname": "Test:hist1:median", "low": 450, "high": 550, "lolo": 420, "hihi": 580}
      ]
    }
  ]
}
```

JSON Schema



online editor

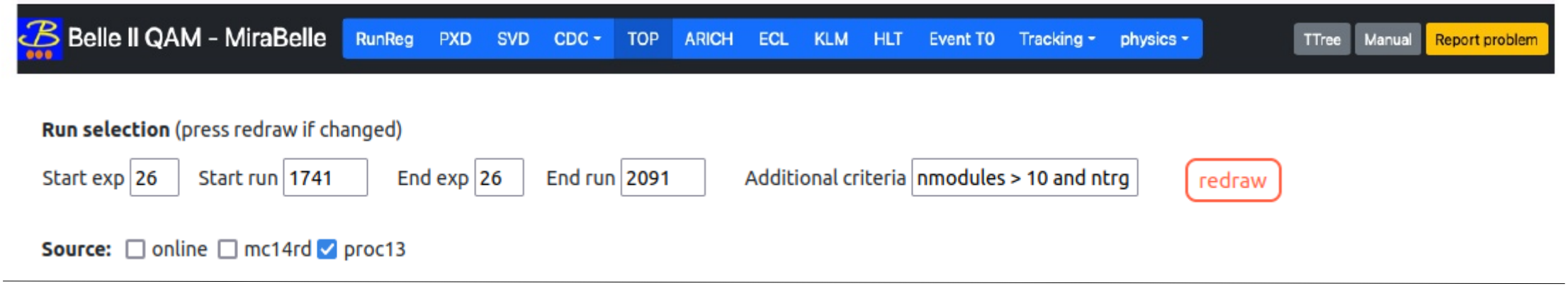
Framework Improvements – Code Cleanup & Development

- Pull requests to dqm package since April (newest first)
  - Bold text indicate framework updates and cleanups

Merge pull request #1357 in B2/basf2 from feature/BII-9613-dqm-pxd-reduction to main
Merge pull request #1341 in B2/basf2 from feature/dqm-fix-doc to main
Merge pull request #1358 in B2/basf2 from feature/BII-9405-grid-lines-KLM-plots to main
Merge pull request #1356 in B2/basf2 from feature/BII-9616-pxd-dqm-read-epics-limits-from-pv to main
Merge pull request #1342 in B2/basf2 from feature/BII-9455-verify-class to main
Merge pull request #1354 in B2/basf2 from feature/BII-9613-dqm-ip-monitoring-test to main
Merge pull request #1344 in B2/basf2 from feature/BII-9613-dqm-update-example-test-code to main
Merge pull request #1353 in B2/basf2 from feature/BII-9613-dqm-image-export to main
Merge pull request #1352 in B2/basf2 from bugfix/fix-dqmhistanalysisklm-module to main
Merge pull request #1343 in B2/basf2 from feature/BII-9613-dqm-update-ip-monitoring-delateHisto to main
Merge pull request #1250 in B2/basf2 from feature/BII-9407-monitor-klmeff-in-mirabelle to main
Merge pull request #1280 in B2/basf2 from feature/BII-9546-histogra-update-check to main
Merge pull request #1281 in B2/basf2 from feature/BII-9546-add-underflow-mode to main
Merge pull request #1218 in B2/basf2 from feature/BII-9546-change-in-base-class to main
Merge pull request #1251 in B2/basf2 from feature/BII-9548-cleanup-pxd to main
Merge pull request #1244 in B2/basf2 from feature/BII-9548-cleanup-includes to main
Merge pull request #1220 in B2/basf2 from feature/BII-9548-cleanup-find-functions to main
Merge pull request #1243 in B2/basf2 from feature/BII-9454-klmdqm2-cosmics to main
Merge pull request #1217 in B2/basf2 from feature/BII-9548-code-quality-and-removal-of-dependencies to main
Merge pull request #1216 in B2/basf2 from feature/BII-9547-remove-unused-histmemory-class-from-dqm-package to main
Merge pull request #1015 in B2/basf2 from feature/BII-9324-dqmhistanalysisklm-error-scaling to main
Merge pull request #1045 in B2/basf2 from feature/change_limit_taganadqm_maskbb2_main to main
Merge pull request #1107 in B2/basf2 from feature/add-ecl-monobj to main
Merge pull request #1096 in B2/basf2 from feature/BII-7666-remove-nsm-dependence-in-dqm to main
Merge pull request #1063 in B2/basf2 from feature/dqm_pxd_add_debug_output to main
Merge pull request #1055 in B2/basf2 from feature/dqm_pxd_add_debug_output to main
Merge pull request #1048 in B2/basf2 from feature/BII-9389-updating-klmdqm2-parameters-for-bb2 to main
Merge pull request #982 in B2/basf2 from bugfix/BII-9351-exclude-bad-tdc-adc-in-calculating-the-mean-of-tdc-slope-adc-mean to main
Merge pull request #991 in B2/basf2 from feature/BII-9360 to main
Merge pull request #994 in B2/basf2 from feature/BII-9361-parameters-change-of-klmdqm2-in-commondqm to main
Merge pull request #937 in B2/basf2 from feature/dqm_analysis_fix_mean to main

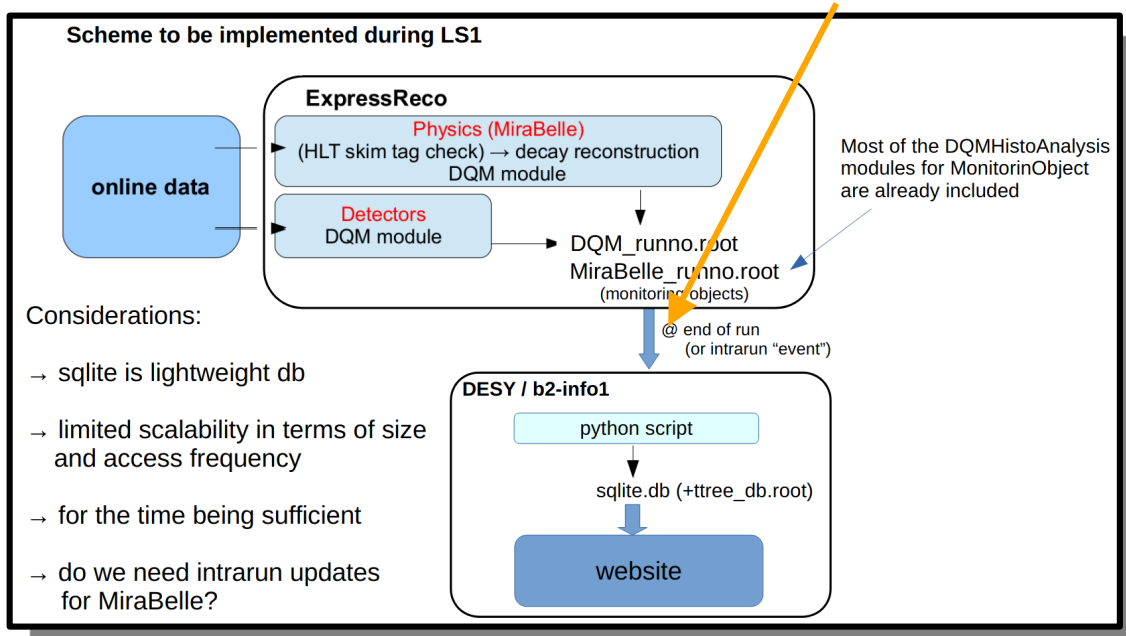
```
git diff --shortstat "@{2 month ago}" dqm
 105 files changed, 2108 insertions(+), 1526 deletions(-)
```

MiraBelle – Trend Plotting



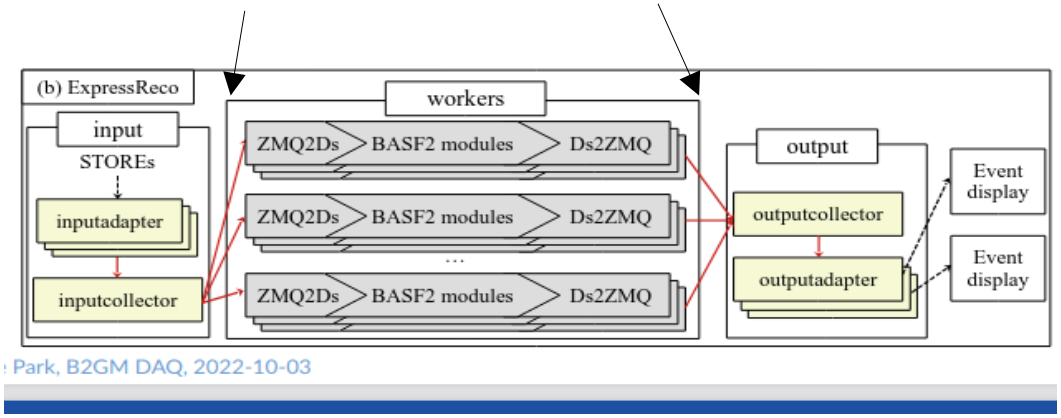
- By now all detectors added content
- Todo/ongoing
  - Stream-line operation
    - Skipping kekcc processing step
  - Still discussion point: Intra-run updates
  - Review what is shown per detector (sensitivity to problems?)
- One core developer, detectors responsible for adding & displaying their content

Run directly on DQM server instead of later on kekcc



ExpressReco: Recent Updates & Plans

- ERECO → ZeroMQ based (like HLT)
  - Expect increase reliability (as already shown for HLT)
  - Should get rid of histogram content "bleeding" at run change

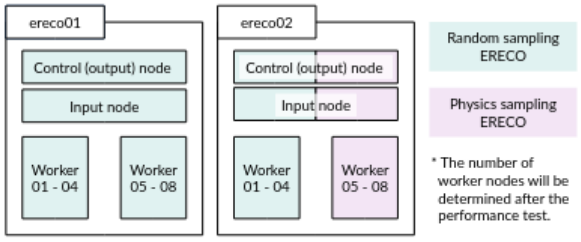


Park, B2GM DAQ, 2022-10-03

- ERECO only processes fraction of events → lower statistics
- Want better **physics channel/observable** monitoring
  - Introducing dedicated ERECO nodes for physics skims
  - Keep minimum bias events for detector monitoring

Dedicated physics ERECO

- Performance study for trigger line preparation is being continued by Daniel
  - If we use D^* skim only, one worker node is enough
- Remaining tasks before/after using HLT01-05
  - Writing config files for ERECO01
  - HLT (STORE) inclusion/exclusion checking while loading (need multiple STOREs)
  - Event display connection
  - Additional discussion with Longke, after generating physics DQM files in ERECO01

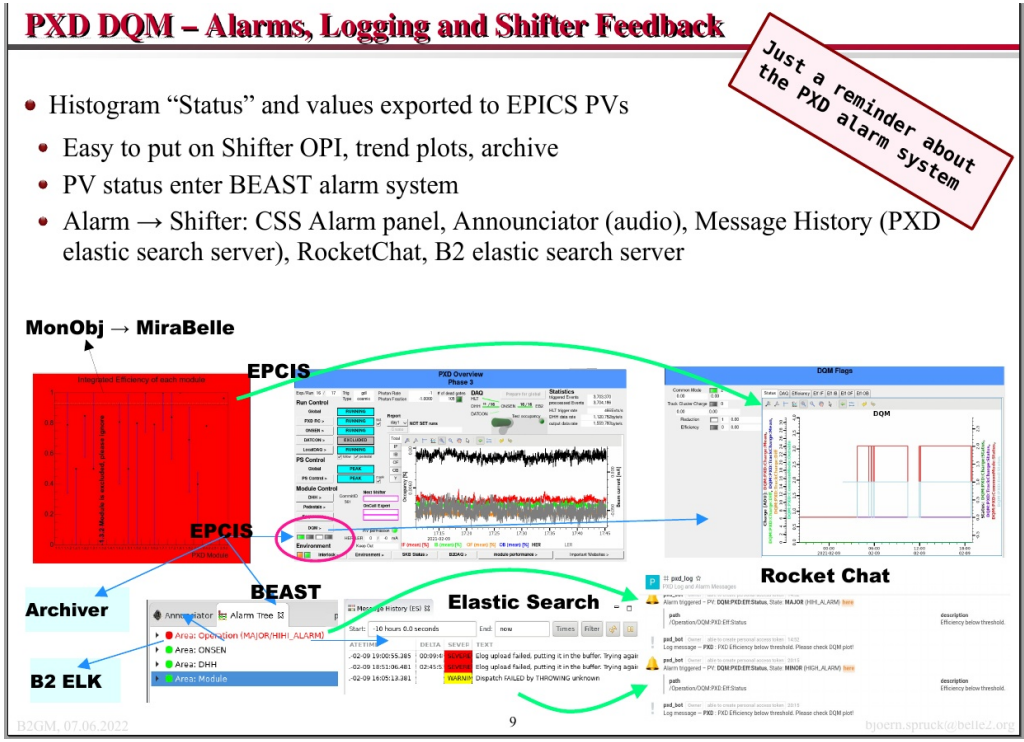


Seokhee Park, B2GM DAQ, 2022-10-03

Ongoing Framework Updates – Intra-Run Monitor/Delta/Alarms

- Use LS1 for rework the framework (bug-fixing, code clean up and simplify)
  - Add new features
- Time sliced “delta” histogramming necessary to see fast changes in long runs (mitigate run-integrated histograms) → used since 2019/20 for PXD and IP monitoring
  - Implemented recently within DQM framework for all analysis modules (as option)
    - Configurable per histogram
  - Allows for smarter monitoring and detection of anomalies , e.g. possibility to overlay/comparing last n delta histograms

- Simplify the use of EPICS for export to monitor/alarm for shifter
- Plan: use of configuration file to simplify reuse of analysis (parameters), test cases, EPICS definitions (alarm limits, messages etc), …



Vision … Where Do We Want to Go?

- Seen from the detector side, what do they want?

```
Histogram → Observable → Alarm Limits
```

- They do not want to write **any** code nor want to know the technical details! Abstraction is needed; high level configurations are the key!

```
MyDetectorTrend={
    HistogramName: “histogram name”
    DeltaBy: “time”/”event”/”entries”
    ExtractObservable: “mean”/”fit function”
    PVName: “name of the EPICS variable to create”
    AlarmDefinition: Limit,Message,type
    . . .
}
```

- DQM Framework is supposed to extract/interpret this
  - Define the basf2 analysis script with parameters
  - Run/update the IOCs
  - Register them at the archiver
  - …

This would need active development beyond our current person-power. But mainly it is putting together **existing** puzzle pieces.

Misc Updates: Discussion Points

- Common look, layout, behavior (best practice → confluence)
- Identical naming of "common" parameters and variables.
- "private" reference files (and folder structure within)
  - Suggestion: Must use the same folder structure as the histogram filler
- Own code development vs editing python scripts vs JSON configuration files
  - e.g. delta histogramming, EPICS, alarm handling
  - JSON schema for configuration file ← comments requested
  - Early adopters and feed-back welcome

JSON Schema

Clean-up, Bugfixes, Documentation

- Removal of **daq package dependence** where ever possible, thus can run local test now with default basf2 setup.
  - Re-shuffled directory structure of source code.
  - Only DqmInput from Shared Memory has a daq dependence still, but will only be compiled if daq package is available
  - StringSplit (=tokenizer) replaced with a newly implemented one in base class.
- Code quality
  - Some proper checks on size of arrays before usage added
- Removal of old **unused classes and modules** (what to do with “example” codes?)
- Removed **unneeded includes** (even more after changes in base class)
- Documentation
  - **Several modules had completely wrong description** (copy-n-pasted) → fixed
  - Doxygen checks only existence of documentation, not the content. Please be more careful when approving pull requests.
- Changed several B2INFO into B2DEBUG
  - → there are many more which do not yield any useful information! Log file size issue.

Technical Details (Update Histogram/Canvas Flagging)

- Keep a map of histogram and their previous state. Flag histograms at input if they deviate from stored state. (→ vector was replaced by a map of objects)
  - By default, user code does not see any difference, minor modification is needed
  - Change: call findHist with additional parameter “onlyUpdated”
  - For delta histograms, this will be the default from the beginning

```
35      -   typedef std::map<std::string, TH1*> HistList;
        36 + typedef std::map<std::string, HistObject> HistList;
36      37  /**
```

Histogram update tagging

- Keep a map of canvases and their flag.
  - In the default setting, no change to current behavior
  - User code should call a function to explicitly state if canvas has been updated or not.
  - Output code checks the flag and behaves accordingly
    - → future: when all user code has changed, the default behavior of output code can be adapted and we can remove the iteration over in-memory

Canvas update tagging

```
30   30   DQMHistAnalysisModule::HistList DQMHistAnalysisModule::g_hist;
31   31   DQMHistAnalysisModule::MonObjList DQMHistAnalysisModule::g_monObj;
32   32   DQMHistAnalysisModule::DeltaList DQMHistAnalysisModule::g_delta;
     33 + DQMHistAnalysisModule::CanvasUpdatedList DQMHistAnalysisModule::g_canvasup;
33   34
```

```
/**
 * Mark canvas as updated (or not)
 * @param name name of Canvas
 * @param updated was updated
 */
void UpdateCanvas(std::string name, bool updated = true);
```

```
// get only updated histograms
auto hist=findHist(histdir,histname,true);
if(hist==nullptr){
    UpdateCanvas(m_canvas->getName(), false);
}else{
    // update canvas m_canvas e.g.
    m_canvas→cd();
    Hist->Draw(“hist”);
    UpdateCanvas(m_canvas->getName(), true);
}
```

Small changes to user code needed:
(only schematic to show the concept)

Details

- Keep a map of histogram and their previous state. Flag histograms at input if they deviate from stored state. (→ vector was replaced by a map of objects)
  - By default, user code does not see any difference, minor modification is needed
  - Change: call findHist with additional parameter “onlyUpdated”
  - For delta histograms, this will be the default from the beginning

Histogram update tagging

```
35      -    typedef std::map<std::string, TH1*> HistList;
     36 +    typedef std::map<std::string, HistObject> HistList;
36   37     /**
```

- Keep a map of canvases and their flag.

- In the default setting, **no change to current behavior**
- User code should call a function to explicitly state if canvas has been updated or not.
- Output code checks the flag and behaves accordingly
  - → future: when all user code has changed, the default behavior of output code can be adapted and we can remove the iteration over in-memory

Canvas update tagging

```
30   30    DQMHistAnalysisModule::HistList DQMHistAnalysisModule::g_hist;
31   31    DQMHistAnalysisModule::MonObjList DQMHistAnalysisModule::g_monObj;
32   32    DQMHistAnalysisModule::DeltaList DQMHistAnalysisModule::g_delta;
     33 +  DQMHistAnalysisModule::CanvasUpdatedList DQMHistAnalysisModule::g_canvasup;
33   34
```

```
/**
 * Mark canvas as updated (or not)
 * @param name name of Canvas
 * @param updated was updated
 */
void UpdateCanvas(std::string name, bool updated = true);
```

```
// get only updated histograms
auto hist=findHist(histdir,histname,true);
if(hist==nullptr){
    UpdateCanvas(m_canvas->getName(), false);
}else{
    // update canvas m_canvas e.g.
    m_canvas→cd();
    Hist->Draw(“hist”);
    UpdateCanvas(m_canvas->getName(), true);
}
```

Small changes to user code needed:
(only schematic to show the concept)

Technical Implementation ("Delta")

https://agira.desy.de/browse/BII-9546

- Add a new list of objects to keep track of delta settings (per histogram) and actual histograms
  - Configurable update mechanism
- Functions to register parameters and request histograms

Delta Histogram base class changes

```
30  30  DQMHistAnalysisModule::HistList DQMHistAnalysisModule::g_hist;
31  31  DQMHistAnalysisModule::MonObjList DQMHistAnalysisModule::g_monObj;
    32 + DQMHistAnalysisModule::DeltaList DQMHistAnalysisModule::g_delta;
    33 +
```

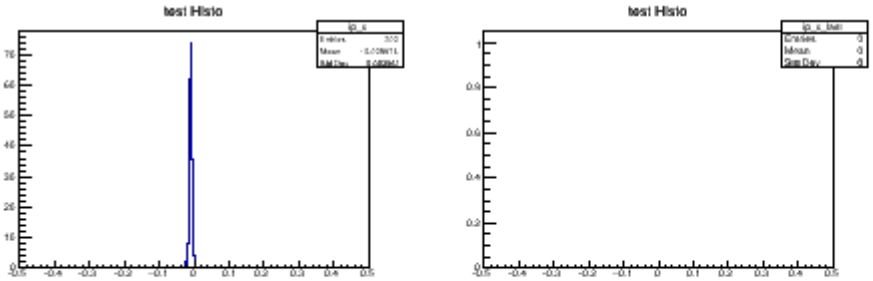
```
156 +    /**
157 +     * Get Delta histogram.
158 +     * @param dirname directory
159 +     * @param histname name of histogram
160 +     * @param n index of delta histogram, 0 is most recent one
161 +     * @return delta histogram or nullptr
162 +     */
163 +    TH1* getDelta(const std::string& dirname, const std::string& histname, int n = 0);
164 +
165 +    /**
166 +     * Add Delta histogram parameters.
167 +     * @param dirname directory
168 +     * @param histname name of histogram
169 +     * @param t type of delta histogramming
170 +     * @param p numerical parameter depnding on type, e.g. number of entries
171 +     * @param a amount of histograms in the past
172 +     */
173 +    void addDeltaPar(const std::string& dirname, const std::string& histname, int t, int p, unsigned int a);
174 +
```

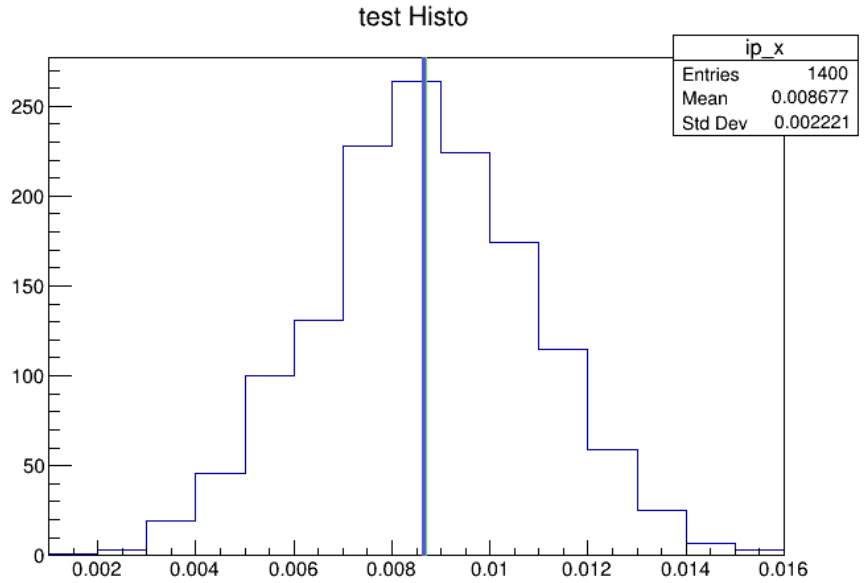
```
41 +    /**
42 +     * The type of list of delta settings and histograms.
43 +     */
44 +    typedef std::map<std::string, HistDelta*> DeltaList;
45
```

```
16 + /**
17 +  * Class to keep track of delta histograms
18 +  */
19 + class HistDelta {
20 + public:
21 +   int m_type; /**< type of delta algo, 0=disable */
22 +   int m_parameter; /**< parameter depending on algo, e.g. nr of entries or events */
23 +   unsigned int m_amountDeltas; /**< amount of past histograms, at least 1*/
24 +   TH1* m_lastHist;/**< Pointer to last histogram state for check */
25 +   std::vector<TH1*> m_deltaHists;/**< vector of histograms (max m_amountDeltas) */
26 + public:
27 +
28 +   /** Construktor
29 +    * @param t type
30 +    * @param p parameter for type
31 +    * @param a amount of deletas in the past
32 +    */
33 +   HistDelta(int t = 0, int p = 0, unsigned int a = 0);
34 +
35 +   /** Parameter setter
36 +    * @param t type
37 +    * @param p parameter for type
38 +    * @param a amount of deletas in the past
39 +    */
40 +   void set(int t, int p, unsigned int a);
41 +
42 +   /** Check if update of delta histogram is necessary
43 +    * @param hist pointer to histogram
44 +    */
45 +   void update(TH1* hist);
46 +
47 +   /** Reset histogram and deltas, not the parameters
48 +    */
49 +   void reset(void);
50 +
51 +   /** Get Delta Histogram
52 +    * @param n number of delta into teh past, 0 is most recent one
53 +    * @return Found histogram or nullptr
54 +    */
55 +   TH1* getDelta(unsigned int n = 0);
56 + };
```

IP Position Monitoring

- IP position monitoring was first to adapt to new framework function as can be seen as a simple example, even including EPICS PV integration
- Code for testing added (b2test), configured by json config file





Reminder: Code-free Histogram Analysis

- Idea: Adding simple histogram analysis without writing any code
  - We already had this in mind when writing the first example analysis modules in 2017
  - → DQMHistComparitor (compare a list histograms against its references and color canvas accordingly)
    - Chi2 and Kolmogorov test are very strict tests, cut tuning per histogram needed
  - → DQMHistAnalysisEpicsExample
    - Supply histogram and fit function, result to EPICS
    - Maybe too basic to be directly usable
- (Those modules have not been used and therefore not been updated for a while.)
- Not sure whats better, one module per histo or one module processing a list, both possible

Simply add another module instance to python script

```
# Create main path
main = b2.create_path()

# Modules
input = b2.register_module('DQMHistAnalysisInput')
input.param('HistMemoryPath', argv[1])
main.add_module(input)

main.add_module('DQMHistAnalysisExample',HistoName= "DAQ/h_Gaus_2",
Function= "gaus(0)")
main.add_module('DQMHistAnalysisExample',HistoName= "DAQ/h_Other_1",
Function= "gaus(0)")

output = b2.register_module('DQMHistAnalysisOutputRelayMsg')
output.param("Port", 9192)
main.add_module(output)

# Process all events
b2.process(main)
```

DQMHistAnalyserCheckRefHistoFile.py

Date: Wed Dec 13 16:33:46 2017 +0100

```
# Create main path
main = b2.create_path()

# Modules
input = b2.register_module('DQMHistAnalysisInput')
input.param('HistMemoryPath', argv[1])
input.param('AutoCanvas', False)
main.add_module(input)

checker = b2.register_module('DQMHistComparitor')
checker.param('HistoList', [
    ['FirstDet/h_HitXPositionCh01', 'ref/FirstDet/h_HitXPositionCh01', 'xdet/test1', '100',
    '0.9', '0.6', 'test1'],
    ['FirstDet/h_HitYPositionCh01', 'ref/FirstDet/h_HitYPositionCh01', 'ydet/test2', '100',
    '0.9', '0.6', 'test2'],
    ['SecondDet/h_HitCh01', 'ref/SecondDet/h_HitCh01', 'test3', '100', '0.9', '0.6', 'test3']
]
)
main.add_module(checker)
```

Simply add one line to python script

Reminder: Code-free Histogram Analysis

- Idea: Adding simple histogram analysis without writing any code
  - We already had this in mind when writing the first example analysis modules in 2017
  - → DQMHistComparitor (compare a list histograms against its references and color canvas accordingly)
    - Chi2 and Kolmogorov test are very strict tests, cut tuning per histogram needed
  - → DQMH
    - Supply hi
    - Maybe to
- (Those mod
- Not sure wh possible

They may be a better way.
… by configuration files!
Implementation currently ongoing …

```
# Create main path
main = b2.create_path(

# Modules
input = b2.register_module('DQMHistAnalysisInput')
input.param('HistMemoryPath', argv[1])
main.add_module(input)

main.add_module('DQMHistAnalysisExample',HistoName= "DAQ/h_Gaus_2",
Function= "gaus(0)")
main.add_module('DQMHistAnalysisExample',HistoName= "DAQ/h_Other_1",
Function= "gaus(0)")

output = b2.register_module('DQMHistAnalysisOutputRelayMsg')
output.param("Port", 9192)
main.add_module(output)

# Process all events
b2.process(main)
```

```
efHistoFile.py
16:33:46 2017 +0100
# Modules
input = b2.register_module('DQMHistAnalysisInput')
input.param('HistMemoryPath', argv[1])
input.param('AutoCanvas', False)
main.add_module(input)

checker = b2.register_module('DQMHistComparitor')
checker.param('HistoList', [
    ['FirstDet/h_HitXPositionCh01', 'ref/FirstDet/h_HitXPositionCh01', 'xdet/test1', '100',
    '0.9', '0.6', 'test1'],
    ['FirstDet/h_HitYPositionCh01', 'ref/FirstDet/h_HitYPositionCh01', 'ydet/test2', '100',
    '0.9', '0.6', 'test2'],
    ['SecondDet/h_HitCh01', 'ref/SecondDet/h_HitCh01', 'test3', '100', '0.9', '0.6', 'test3']
]
)
main.add_module(checker)
```

Simply add one line to python script

bjoern.spruck@belle2.org

Clean-up, Bugfixes, Documentation

- Removal of daq package dependence where ever possible, thus can run local test now with default basf2 setup.
  - Re-shuffled directory structure of source code.
  - Only DqmInput from Shared Memory has a daq dependence still, but will only be compiled if daq package is available
  - StringSplit (=tokenizer) replaced with a newly implemented one in base class.
- Code quality
  - Some proper checks on size of arrays before usage added
- Removal of old unused classes and modules (what to do with “example” codes?)
- Documentation
  - Several modules had completely wrong description (copy-n-pasted) → fixed
  - Doxygen checks only existence of documentation, not the content. Please be more careful when approving pull requests.
- Changed several B2INFO into B2DEBUG
  - → there are many more which do not yield any useful information!

Code Cleanup: Identical Naming etc

- Suggest or enforce identical naming where applicable
  - Parameters e.g.:
    - “HistDir” for histogram directory (or histoDir, histoDirectory, histDirName ...)
    - “HistName” for name (excluding directory prefix)
  - (member) Variables
    - m_histDir
    - m_histName
    - m_canvas
  - Same for common things like monitoringObject and EPICS …

Q/R

What is the delay between data taking and having plots avaialble ? Is this determined by processing time, or time to obtain meaningfull statistics ?

Not simple to answer as several things play a role here:
Overall, it is not determined by the processing time (reconstruction and filling) but by the interval in which the histograms are added up. This is 30s on HLT but only 1000s(!) on the ERECO (due to different framework). Different thread update at different times, thus a steady update of histograms should happen from the beginning (but sometimes is not). On top of that ERECO only start processing events ~2min after run start. The time needed for analyzing the histograms is rather negligible.
(the transport/adding up in the different hierarchies is trigger with a certain time interval: 30s(HLT), 100-120s(ERECO).

For HLT: Updates happen within first minute after run start. Statistics is not an issue as ALL events are processed.
For ERECO: Updates happen at least 2min later, and long delays (>15min) have been observed for some runs (for which the reason has not completely understood). ERECO processes only a limited number of events, thus it takes longer to get statistics for some specific plots.

The switch to ZeroMQ together with the rework of the storage process should give us a significant improvement on the ERECO. Statistic for physics channels should be increasing faster due to use of physics skims.

The adding is done in a single place for all producers? Or is there a hierarchy? How is the load distributed over several DQM servers?

There is a hierarchy. First all hist per worker node are added, then all worker nodes per unit, and finally all units together. Only the last step happens on the dqm server.

What fraction of the data will be processed on-line for performance tests?

Not sure if I understand the question correctly: During normal operation, all event are processed (thus monitored) on HLT, but ERECO is limited by computing power and more complex reconstruction to around 500-1000 events/s.

Concerning review:

how will this review be organised? What is the timescale ?

This request came up only recently when summarizing the progress for the last collaboration meeting. There is no plan yet. But clearly (internal) reviewers from outside the actual detector groups should be involved.

This makes sense to me!
Are you planning have expert shifters for all detecter subsystems? probably initially. But the information should be available remotely, and shifters on day shifts at home could analyze the various monitoring information and contact experts as needed!

Overall, the goal should be to reduce the load on the sub-system experts by having a better monitoring presented to the control room (or remote) shifters.

What is the relation between EPCIS and "DCS"/"Slow Control" in the presentation of Kunigo ?

Ideally, the output of DQM analysis would enter the alarm and messaging system of the overall detector control and monitoring system. (EPICS and NSM2 are the two underlying slow control/monitoring protocols used by BelleII and SuperKEKB.)

At the moment, this is done under detector responsibility: f.e. ECL alarm goes directly to ECL shifter. PXD alarm and monitoring enter PXD monitoring and alarm system (and from there to shifter). IP position is routed by PXD servers to SuperKEKB.

How big is the maintenance effort for the current system? How much effort is estimated to be needed for the rework during LS1?

Hard to sum up as different people are involved from different groups (DAQ, DQM, detectors), no complete picture.

Software updates during maintenance days take several hour of time (every two weeks). Updates to web page etc. need support attention but not much time. During operation, mostly no time is needed beside, someone need to be reachable at all times.
Restarting copy file processes now and then, important for on-line (call!) but less critical for MiraBelle.

For LS1 activities, it depends on the priorities we want to set. Works is divided, detector groups (plots and analysis) vs DQM framework and operation.