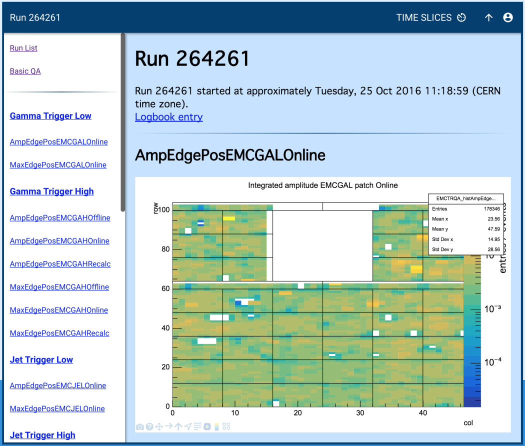


Run 264261 TIME SLICES

Run List

Basic QA

Gamma Trigger Low

AmpEdgePosEMCGALOnline

MaxEdgePosEMCGALOnline

Gamma Trigger High

AmpEdgePosEMCGAHOffline

AmpEdgePosEMCGAHOnline

AmpEdgePosEMCGAHRecalc

MaxEdgePosEMCGAHOffline

MaxEdgePosEMCGAHOnline

MaxEdgePosEMCGAHRecalc

Jet Trigger Low

AmpEdgePosEMCJELOnline

MaxEdgePosEMCJELOnline

Jet Trigger High

Run 264261

Run 264261 started at approximately Tuesday, 25 Oct 2016 11:18:59 (CERN time zone).
Logbook entry

AmpEdgePosEMCGALOnline

Integrated amplitude EMCGAL patch Online

EMCTRQA_histAmpEdge...	
Entries	178346
Mean x	23.56
Mean y	47.59
Std Dev x	14.95
Std Dev y	28.56

ALICE OVERWATCH

Online detector monitoring and basic QA via the HLT

Raymond Ehlers[1], James Mulligan[1]

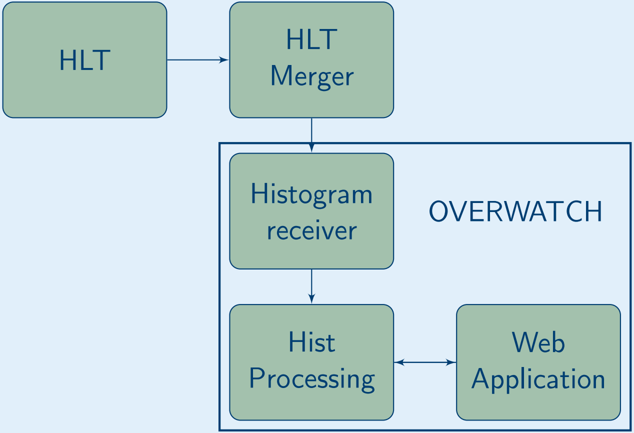
[1]Yale University

Oct 26, 2016

ALICE OVERWATCH

- Provides the processing and interface for online (expert) detector monitoring and basic QA using data from the HLT.
- It began as a project to provide online real-time feedback on the EMCal during the 2015 PbPb run.
    - Has since expanded to support additional detectors with additional features.
- OVERWATCH handles spectra, 2D histograms (for example, EMCal cell amplitudes), etc.
- Code available at: `https://github.com/raymondEhlers/OVERWATCH`

OVERWATCH Architecture



HLT

HLT Merger

Histogram receiver

OVERWATCH

Hist Processing

Web Application

OVERWATCH Architecture

- OVERWATCH is python and ROOT based.
- Split into two main parts:
  - Processing
  - The Web App
    - Depends on the processing module for user requested processing.
- Receiver from HLT written in C++ and utilizes ZMQ.
  - Data is received approximately every minute and time stamped.
  - Large, but not unreasonably large, amount of volume. ($\approx$ 100 GB/year for EMCal + HLT + $\approx$ 3 months of TPC data).
- Designed to run as micro-service, so can start instances as needed.
- Since OVERWATCH processes data from the HLT, our architecture is similar to data processing for Run 3 when the HLT->Event Processing Node.

OVERWATCH Processing

- Processing utilizes PYROOT and runs every minute on newly received data.
- Manages run and subsystem data via ZODB (Zeo Object Database)
  - Makes management of python objects straightforward.
  - Also used by Indico.
  - We aren't strongly attached to this DB.
    - Any appropriate SQL or NoSQL database would be fine.
    - Code is not really reliant on ZODB - easy to switch elsewhere.
  - Structure is hierarchical.
    - Run->Subsystems->HistogramGroups->Histograms.
- Actual data is just stored on disk in root files.
- Output of processing is stored in json files.

OVERWATCH Processing

- Additional processing available per detector/histogram.
  - Can check values in particular histograms, stack hists, create additional hists to summarize, etc.
  - Can also handles alarms.
- Time slices
  - Can make arbitrary selections in time (0-10 minutes, 5-17, whatever, etc) within a run*
  - Can also select processing options. Hot channel thresholds, scale by number of events, etc.
  - Caches result - only reprocess if absolutely necessary.
- Basic trending support for extracted values. More to come.

* - subject to intrinsic time resolution of HLT of 2 minutes encompassing $\approx$ 3 mins.

OVERWATCH Web App

- The Web App is built on Flask.
  - Interface built using Google's Polymer.
  - Pages are built using the Jinja2 template engine. Each detector can build their own.
  - JSROOT used for presenting histograms, with fall back to static images.
  - Navigation handled by AJAX, with fall back to full page reloads.
- Display histograms according to detector specification.

Demo

- Per detector display, sorting.
- Time slices.
  - Time dependence.
  - Processing options.

Status and Outlook

- Previous version running for almost all of 2016 with few issues.
    - Previous version available at: `https://aliceoverwatch.physics.yale.edu/monitoring`
        - Login information available at: `https://twiki.cern.ch/twiki/bin/view/ALICE/L1TriggerMonitoring`.
- Update ready and currently being rolled out.
- We support a micro-services architecture - should support straightforward scaling.
    - Tested for Web App.
    - Still to be tested for processing, but no known show stoppers.
- In discussion with Offline about them hosting the interface.
- Can test yourself using our docker image. See slide in backup.
- Code available at: `https://github.com/raymondEhlers/OVERWATCH`

Thank You

- Thanks to Salvatore Aiola, Markus Fasel, and the HLT!

Backup

Try it yourself using docker

- Docker image available at `https://hub.docker.com/r/rehlers/overwatch/`.
- Can be tested using the following procedure (to be streamlined - we don't deploy processing like this at the moment).
    - Download test data from: `https://aliceoverwatch.physics.yale.edu/testingDataArchive`.
    - `docker run -it -v data:/overwatch/data -e deploymentOption=devel overwatch /bin/bash`
    - `cd /overwatch && python runProcessRuns.py`
    - `cd deploy && python updateDBUsers.py`
    - `cd /overwatch && python runWebApp.py`
- Still testing some cases - please let us know if you run into any trouble!

OVERWATCH

Online **V**isualization of **E**merging t**R**ends and **W**eb **A**ccessible de**T**ector **C**onditions using the **H**LT

Additional improvements

- Improve time series summary support.