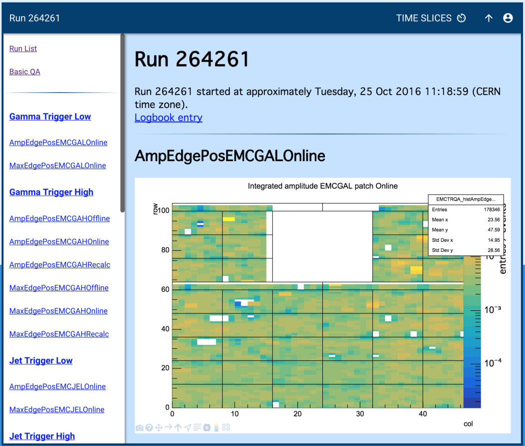

Run 264261

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

ALICE OVERWATCH: Status and Plans

Offline Week - March 2017

Raymond Ehlers on behalf of the OVERWATCH Collaboration

OVERWATCH Collaboration:
Raymond Ehlers (Yale University)
Markus Fasel (ORNL)
Sarah LaPointe (Frankfurt Institute for Advanced Studies)

Mar 28, 2016

Outline

OVERWATCH/1

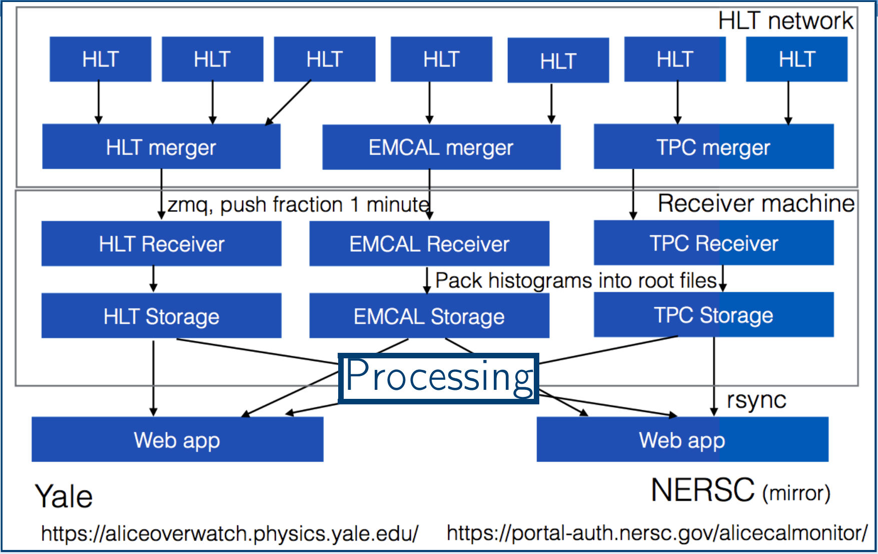
- **OVERWATCH**: Online Visualization of Emerging tRends and Web Accessible deTector Conditions using the HLT.
- Created to **monitor and visualize** QA data from the HLT.
  - Data is stored **persistently**.
- OVERWATCH handles **spectra, 2D histograms** (for example, EMCal cell amplitudes), etc.
- Allows slicing of data in **time windows** ("time slicing").
- Based around two main components:
  - **Processing** based on PyROOT.
  - **WebApp** based on flask.
- Code available at: `https://github.com/raymondEhlers/OVERWATCH`.

OVERWATCH/2

- It began as a project during the 2015 Pb–Pb run to provide online **real-time** feedback on the EMCal (beginning with James Mulligan and RJE).
  - Has since expanded to support **additional detectors and additional features**.
  - Includes a **plug-in framework** to extend nearly any part of the processing.
- Currently supports the **EMCal, TPC, and HLT** (includes assorted histograms without a dedicated HLT merger).
- First version deployed during the 2015 Pb–Pb run, and then ran for all of 2016 with few issues.
- Collected approximately **100 GB** during this period.

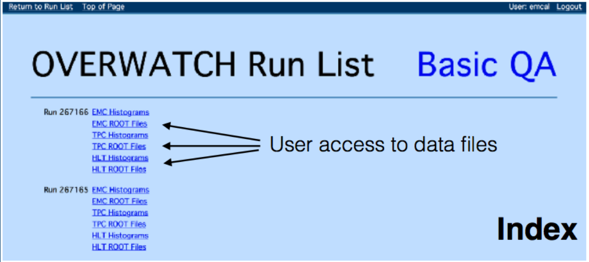
Data Flow from HLT

Image from: M. Fasel



Current Version

Image from: M. Fasel



Return to Run List   Top of Page   User: emcal   Logout

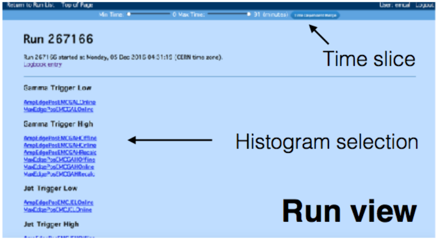
OVERWATCH Run List   Basic QA

Run 267166 EMC Histograms
EMC ROOT Files
TPC Histograms
TPC ROOT Files
HLT Histograms
HLT ROOT Files

Run 267165 EMC Histograms
EMC ROOT Files
TPC Histograms
TPC ROOT Files
HLT Histograms
HLT ROOT Files

User access to data files

Index



Return to Run List   Top of Page   User: emcal   Logout

Min Time: 0 Max Time: 91 (minutes)

Run 267166

Run 267166 started at Monday, 05 Dec 2015 04:31:15 (CERN time zone).
Logbook entry

Gamma Trigger Low

AmpEdgePosEMCGALOnline
MaxEdgePosEMCGALOnline

Gamma Trigger High

AmpEdgePosEMCGAHOffline
AmpEdgePosEMCGAHOnline
AmpEdgePosEMCGAHRecalc
MaxEdgePosEMCGAHOffline
MaxEdgePosEMCGAHOnline
MaxEdgePosEMCGAHRecalc

Jet Trigger Low

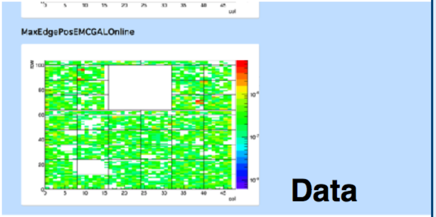
AmpEdgePosEMCJELOnline
MaxEdgePosEMCJELOnline

Jet Trigger High

Time slice

Histogram selection

Run view

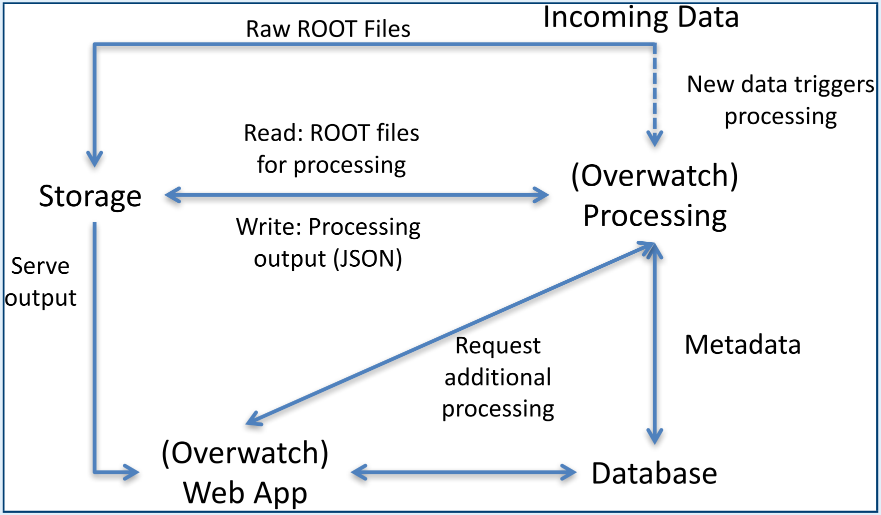


Data

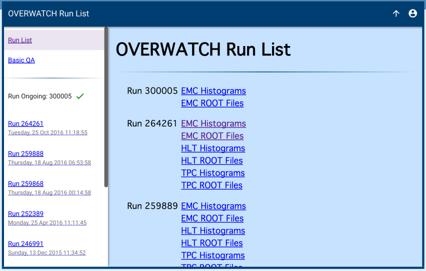
Upgrade to Overwatch

- Major upgrade developed to improve on current version.
- Substantial improvements to user experience.
  - Improved interactivity and visualization with **JSRoot and AJAX**.
  - **Cleaner and more consistent interface** built with Google's Polymer.
- Much more **capable and extensible** back end
  - Much **better performance**.
  - Provides **plug-ins to all stages** of processing.
  - Moves towards micro-services architecture (some improvements still needed here).
- Major infrastructure improvements
  - Migrating **all infrastructure to CERN** by utilizing OpenStack and coordinating with Offline.
  - **Commonly accessible** data storage.
- Hoping to deploy for the beginning of the 2017 pp data taking.

Data Flow in New Version


Incoming Data
Raw ROOT Files
New data triggers processing
Read: ROOT files for processing
Storage
(Overwatch) Processing
Write: Processing output (JSON)
Serve output
Request additional processing
Metadata
(Overwatch) Web App
Database

Next Version Look-and-Feel


OVERWATCH Run List

Run List

Basic QA

Run Ongoing: 300005 ✓

Run 264261
Tuesday, 25 Oct 2016 11:18:55

Run 259888
Thursday, 18 Aug 2016 06:53:58

Run 259868
Thursday, 18 Aug 2016 00:14:58

Run 252389
Monday, 25 Apr 2016 11:11:45

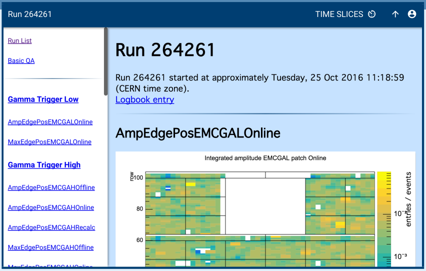
Run 246991
Sunday, 13 Dec 2015 11:34:52

OVERWATCH Run List

Run 300005 EMC Histograms
EMC ROOT Files

Run 264261 EMC Histograms
EMC ROOT Files
HLT Histograms
HLT ROOT Files
TPC Histograms
TPC ROOT Files

Run 259889 EMC Histograms
EMC ROOT Files
HLT Histograms
HLT ROOT Files
TPC Histograms


Run 264261 TIME SLICES

Run List

Basic QA

Gamma Trigger Low

AmpEdgePosEMCGALOnline

MaxEdgePosEMCGALOnline

Gamma Trigger High

AmpEdgePosEMCGAHOffline

AmpEdgePosEMCGAHOnline

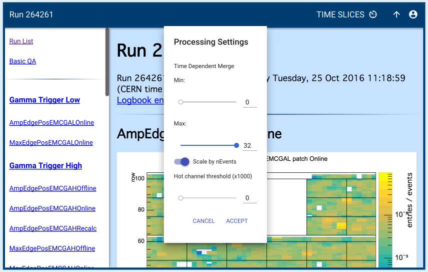
AmpEdgePosEMCGAHRecalc

MaxEdgePosEMCGAHOffline

Run 264261

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

AmpEdgePosEMCGALOnline


Run 264261 TIME SLICES

Run List

Basic QA

Gamma Trigger Low

AmpEdgePosEMCGALOnline

MaxEdgePosEMCGALOnline

Gamma Trigger High

AmpEdgePosEMCGAHOffline

AmpEdgePosEMCGAHOnline

AmpEdgePosEMCGAHRecalc

MaxEdgePosEMCGAHOffline

Processing Settings

Time Dependent Merge

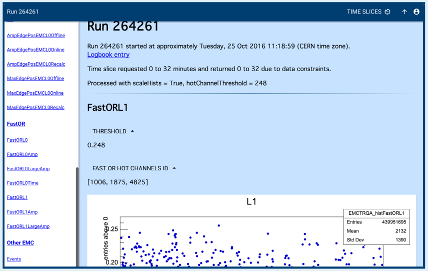
Min: 0

Max: 32

Scale by nEvents

Hot channel threshold (x1000) 0

CANCEL ACCEPT


Run 264261 TIME SLICES

AmpEdgePosEMCL0Offline

AmpEdgePosEMCL0Online

AmpEdgePosEMCL0Recalc

MaxEdgePosEMCL0Offline

MaxEdgePosEMCL0Online

MaxEdgePosEMCL0Recalc

FastOR

FastORL0

FastORL0Amp

FastORL0LargeAmp

FastORL0Time

FastORL1

FastORL1Amp

FastORL1LargeAmp

Other EMC

Events

Run 264261

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

Time slice requested 0 to 32 minutes and returned 0 to 32 due to data constraints.

Processed with scaleHists = True, hotChannelThreshold = 248

FastORL1

THRESHOLD

0.248

FAST OR HOT CHANNELS ID

[1006, 1875, 4825]

EMCTRQA_histFastORL1	
Entries	439951695
Mean	2132
Std Dev	1390

Outline

Deployment and Commission of the Overwatch Upgrade

- Migrated receivers and data to **CERN OpenStack**.
  - Created a **dedicated project** for ALICE HLT QA.
  - Yale site will stay until the transition is completed.
- Data storage on EOS is required for all data to be accessible from the **HLT receivers, processing, and web app**.
  - We now have **dedicated storage** on EOS.
  - The previously collected data was **recently migrated** to EOS.
- Currently adapting some code to properly access data on EOS.
- Data taking is fast approaching and there is still some work to do, but we hope to deploy soon!

Investigation into Elasticsearch for Histogram Storage

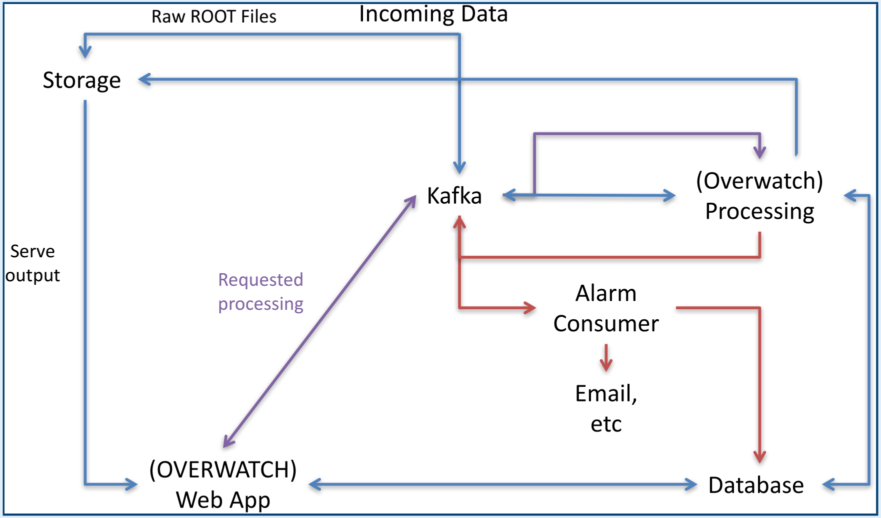
- Overwatch stores full histograms, so we investigated the possibility of storing both metadata and histograms in Elasticsearch.
- Single database source could simplify operation (such as interactively accessing data via Swan).
- Challenging due to the **large amount of redundant histogram information**, amongst other issues.
- Developed a possible data layout (code linked here).
- Early tests observed the Elasticsearch database was $\sim$**10 times larger** than storing raw data.
- Storage of histograms in Elasticsearch **does not** seem to be feasible.
  - **Still useful for trending** information.
  - Perhaps for metadata, **depending on additional external factors**.
- For further information, see Markus' presentation at the February DPG General Meeting: `https://indico.cern.ch/event/611050/#sc-2-2-online-qa`.

Outline

Apache Kafka

- Data processing is currently triggered by receiving data.
  - This aspect of the system is **not particularly flexible**.
- We are interested in using **Apache Kafka** to managing the processing of incoming data.
- Currently **setting up a small Kafka test system** in the Overwatch OpenStack cluster.
  - Initial tests will attempt to receive histograms into the system and then process them.
- Investigating the potential of **utilizing elements** of the **existing Overwatch processing** functionality.

Architecture with Kafka (Proposed)



Towards Triggering Alarms

- There is already some **very basic** alarm code in the Overwatch processing framework.
- Basic examples include:
  - Check if there are **outliers** from a distribution.
  - Check for values above a **threshold**.
- Work is needed to:
  - **Improve and generalize** the processing framework, particularly with respect to **extracting and saving trending information**.
  - Implement **additional processing functions**, such as comparing to reference spectra.
  - Implement **alarms based on the processing functions and trending**.
- **Service task proposed** to address this need, with a focus on implementing processing functions and alarms.
  - Amongst other activities, need to work with some detector experts to determine how best to implement some example alarms.
- Should be completed with an eye towards integration with Kafka.

Conclusions

- Overwatch provides **online monitoring and visualization** of data from the HLT.
  - The data is **stored persistently**.
- The current version has run for **all of 2016** with few issues.
- We are working **towards deploying the upgraded version** soon.
- Continuing to work on:
  - Investigation of **Apache Kafka**.
  - Improving processing and implementing **trending and alarms** framework.

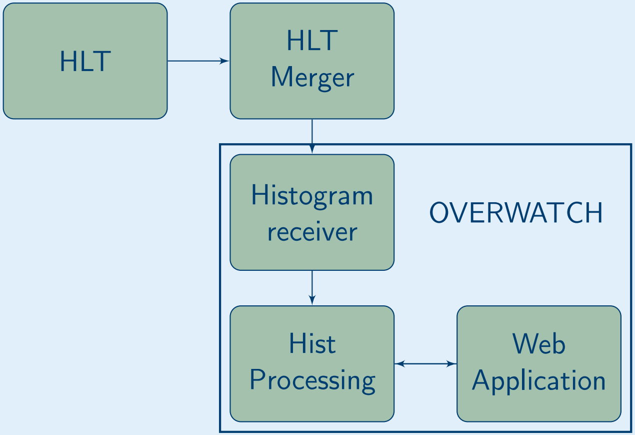
Backup

Service Task Proposal

Development of a trending and alarm framework for online HLT QA

Several QA components on the HLT provide QA data for different detectors at discrete times as simple 1 or multi-dimensional histogram. These data are visualized using the Overwatch web application. The scope of the task is to implement a component processing the histograms and extracting trending information. These trending information will be sent to an Elasticsearch database for visualization. In addition, by comparing to limits defined by users, automatic alarms should be raised (E-Mail to detector responsible with alarm message + Log in database). The alarm handling should be in testing phase during the pp data taking period end of 2017.

OVERWATCH Architecture



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.

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.