ALICE Overwatch time-stamped data

ALICE ML/QC Workshop

Raymond Ehlers[1]

3 December 2018

[1]Relativistic Heavy Ion Group
Department of Physics, Yale University

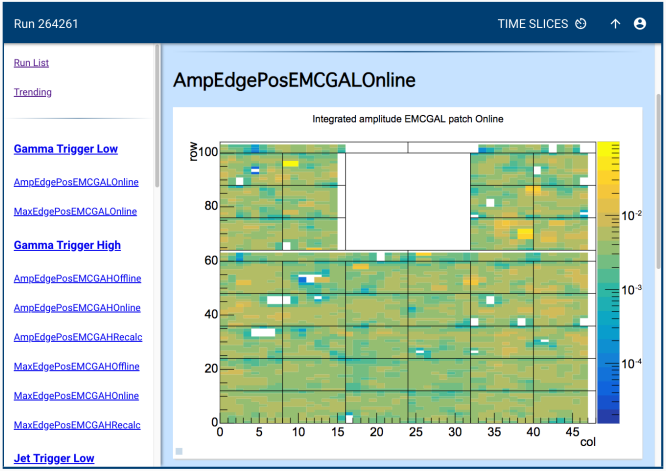
ALICE Overwatch

- **Overwatch**[1] is a project to **monitor and visualize** QA information from the HLT which began in late 2015.
  - Oriented towards expert level information.
  - Complementary to DQM.
  - Available at https://aliceoverwatch.physics.yale.edu.
- Unique capabilities within in ALICE:
  - Monitoring data is stored **persistently**.
  - Data is timed stamped, allowing for slicing of data in **time windows** ("time slicing").
  - Data can be explored via user directed **reprocessing**.

[1] **O**nline **V**isualization of **E**merging t**R**ends and **W**eb **A**ccessible de**T**ector **C**onditions using the **H**LT

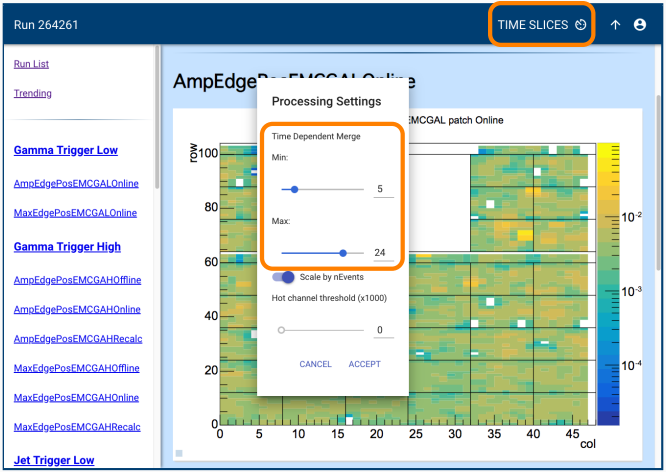
Visualization

- Available at https://aliceoverwatch.physics.yale.edu.



Visualization

- Available at https://aliceoverwatch.physics.yale.edu.

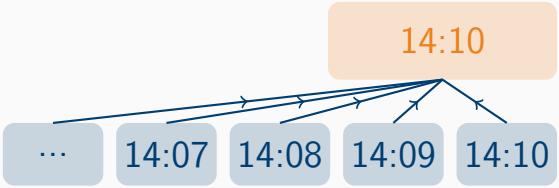


Overwatch Data

- EMC data since November 2015.
  - Includes cluster and cell spectra, triggers info, locations in the detector, timing, etc.
- HLT data since November 2015.
  - Information on HLT performance, tracking performance, etc.
  - Includes other subsystems such as some V0, TPC, ITS histograms.
- TPC data since $\sim$ April 2016.
  - Uses a simplified version of the TPC offline QA code.
  - TPC track spectra, track $\eta - \varphi$, DCA{r,z} vs φ, etc.
- Timestamped ROOT files contain histograms, TObjArray, etc.
  - Provide time slices information across runs.
- Run 2 dataset is approximately 1+ TB.
  - Data volume increased substantially with time.

Using Overwatch Data/1

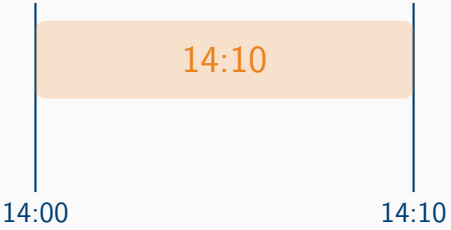
- New file received every minute per subsystem.
- The HLT operates the QA components (which generate the data) on a round-robin best effort basis.
  - The received data is not just data collected during the time between timestamps - it is some subset of the data collected over the last few minutes.



- Depending on the subsystem, the actual time resolution is on the order of 5 mins.
  - Precise granularity depends on QA component, data taking period, HLT details, etc.

Using Overwatch Data/2

- Each data file is cumulative.
  - To get the data received between time n and n+1, one must subtract the histogram, graph, or other object at time n+1 from the object at time n.[2]
- As an example, consider a run starting at 14:00:



[2] See `overwatch.processing.mergeFiles` for further information.

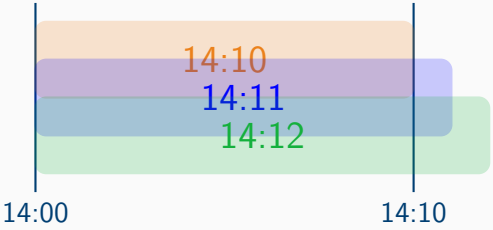
Using Overwatch Data/2

- Each data file is cumulative.
  - To get the data received between time n and n+1, one must subtract the histogram, graph, or other object at time n+1 from the object at time n.[2]
- As an example, consider a run starting at 14:00:

[2] See `overwatch.processing.mergeFiles` for further information.

Using Overwatch Data/2

- Each data file is cumulative.
  - To get the data received between time n and n+1, one must subtract the histogram, graph, or other object at time n+1 from the object at time n.[2]
- As an example, consider a run starting at 14:00:



[2] See `overwatch.processing.mergeFiles` for further information.

Using Overwatch Data/3

- Each timestamp is in the CERN time zone.
  - For a concrete example of handling the time stamps, see `overwatch.utilities.base.extractTimeStampFromFilename`.
  - The `pendulum` python package makes this much easier to handle.
- May have repeated files with updated time stamps appended to the end of a run.
  - Caused by technical details of HLT and Overwatch.
- Resolved by checking whether the number of entries for every object is the same between two files. If so, the newer file can be discarded.
  - Safer to check every object in the file, but just checking the object of interest is often fine.

Accessing Overwatch Data

- Functionality to work with the data is available in the Overwatch package.
  - See the README for more, and `overwatch.processing.moveFiles` for documentation and code examples.
- To access small data volumes:
  - Underlying data files can be accessed directly via the Web App under "ROOT files".
- To access larger data volumes:
  - The unprocessed data is archived on EOS at `/eos/experiment/alice/overwatch`.
    - To access this data, send a request to me and ALICE Offline.
  - REST API file access is also possible under certain circumstances - contact me if this is needed.

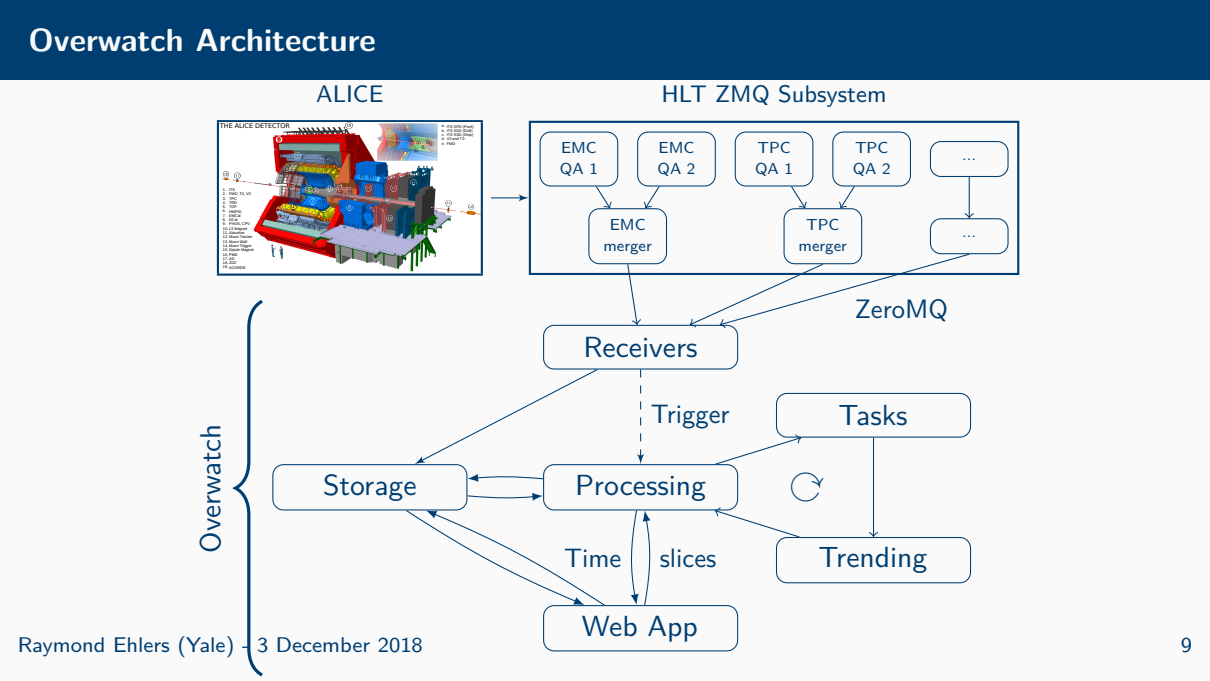
Conclusions

- Overwatch provides monitoring and visualization of data quality using information provided by the ALICE HLT.
  - Timestamped, persistently stored information provides unique capabilities for real-time and post-mortem data exploration.
- Timestamped data is available for training ML models or other data exploration.
  - Time series?
  - Anomaly detection?
- Detector plugin system and data replay available for immediate QC developments.

- Code on GitHub, package on PyPI, container on Docker Hub.

Backup

Overwatch Architecture



Overwatch capabilities

- Receives and stores $\approx$ 300 GB of histograms per year.
  - Increases each year.
- Two main python based components:
  - **Processing** built with `PyROOT`.
  - **WebApp backend** built with `flask`.
- **Front end** built with Google Polymer and JSROOT.
- Processing, trending, and visualization are extensible.
  - Detectors can plug-in to control all aspects of data processing and presentation.