ALICE
A JOURNEY OF DISCOVERY

Dataflow monitoring in ALICE

ALICE, ATLAS, CMS & LHCb Second Joint Workshop on DAQ@LHC

Outline

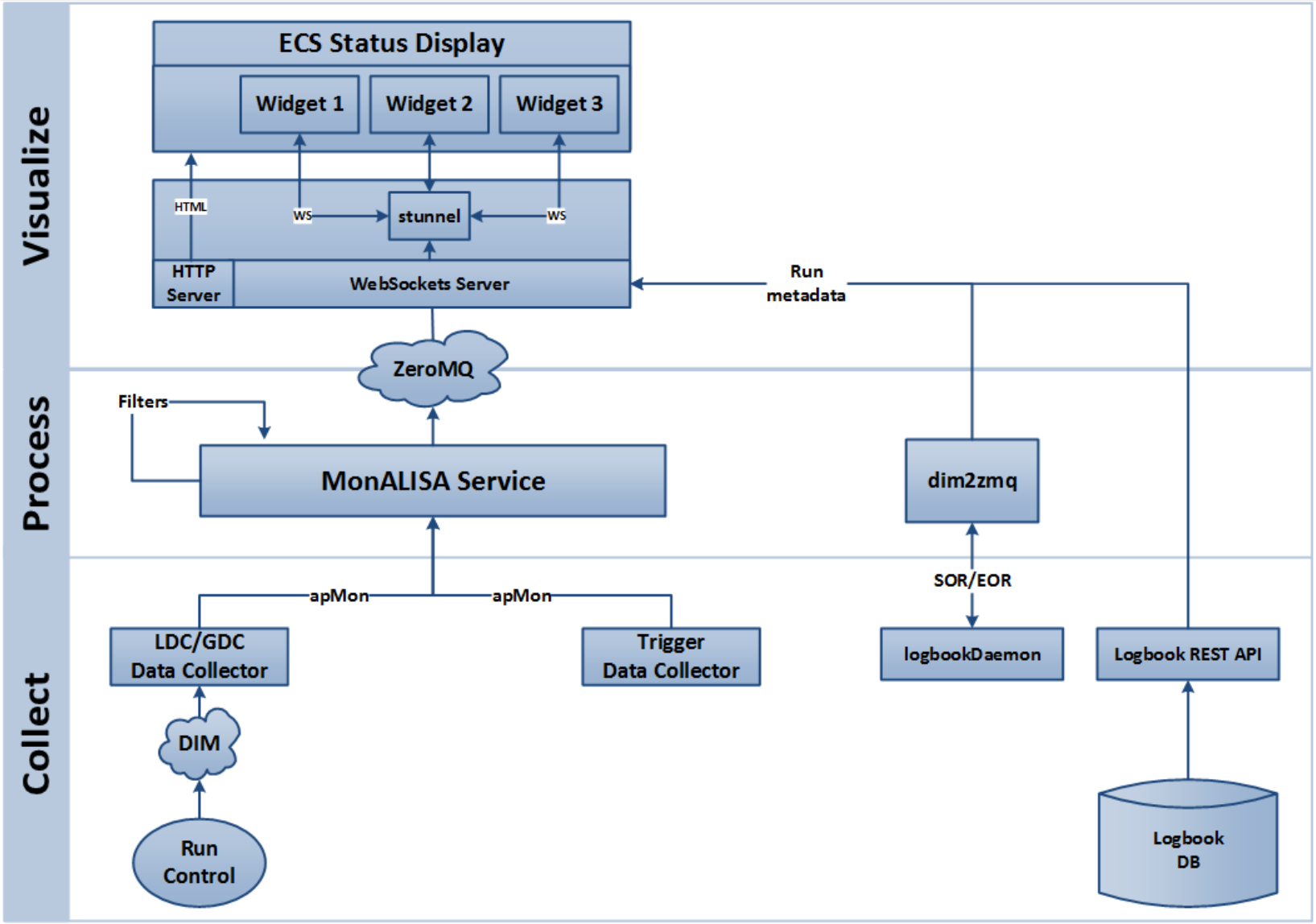
- Motivation & Requirements
- Architecture and components
  - Data Model
  - Data Collectors
  - Data Filters
  - Web Sockets Server
  - Status Display
- Future Work
- Conclusion

Motivation & Requirements

- Discussions during 1st DAQ@LHC Workshop
- Provide shift crews and DAQ oncalls with a centralized view of data flow status
- Reflect operational experience after 3 years of Run 1 operations

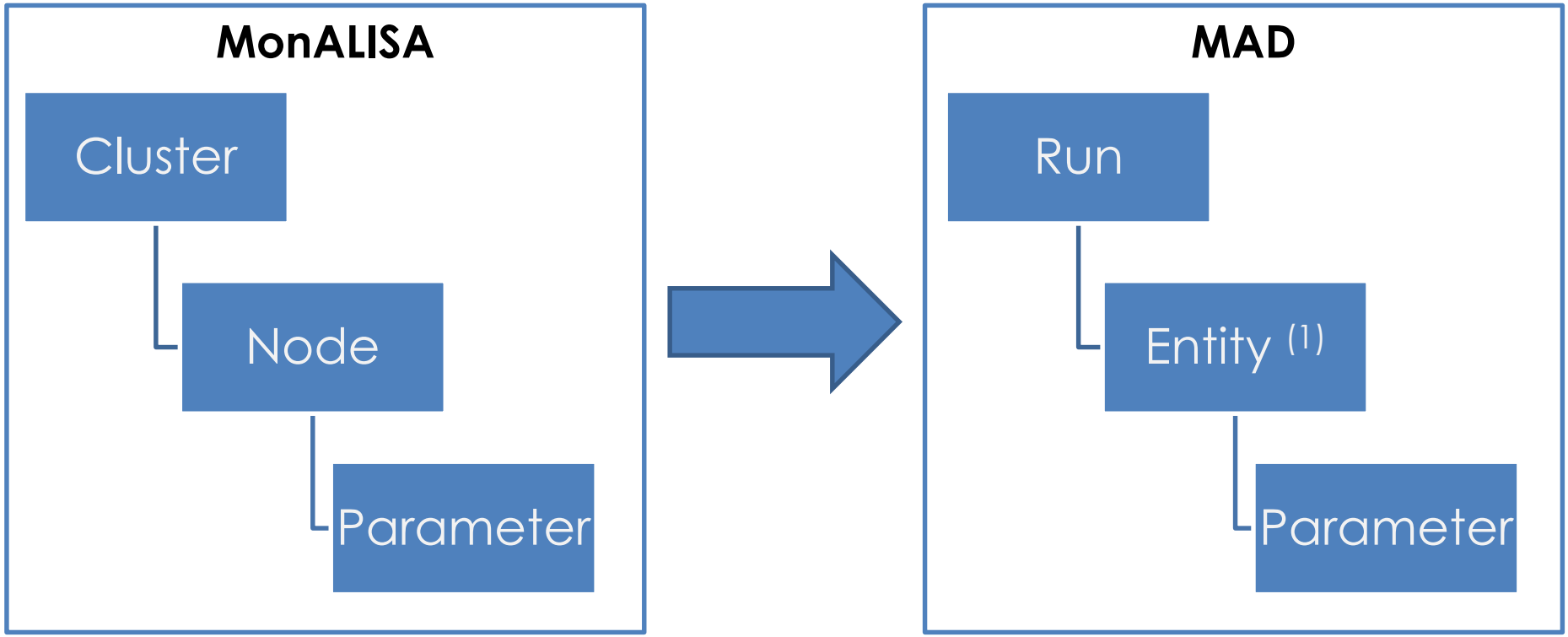
- MAD – Monitoring ALICE Dataflow
  - Negligible impact on existing DAQ machinery
  - Aggregate metrics into higher level values
  - Display should be:
    - Web based, accessible from outside
    - Dynamic, no page refresh
    - Clear content for shift crew

MAD Architecture



Data Model

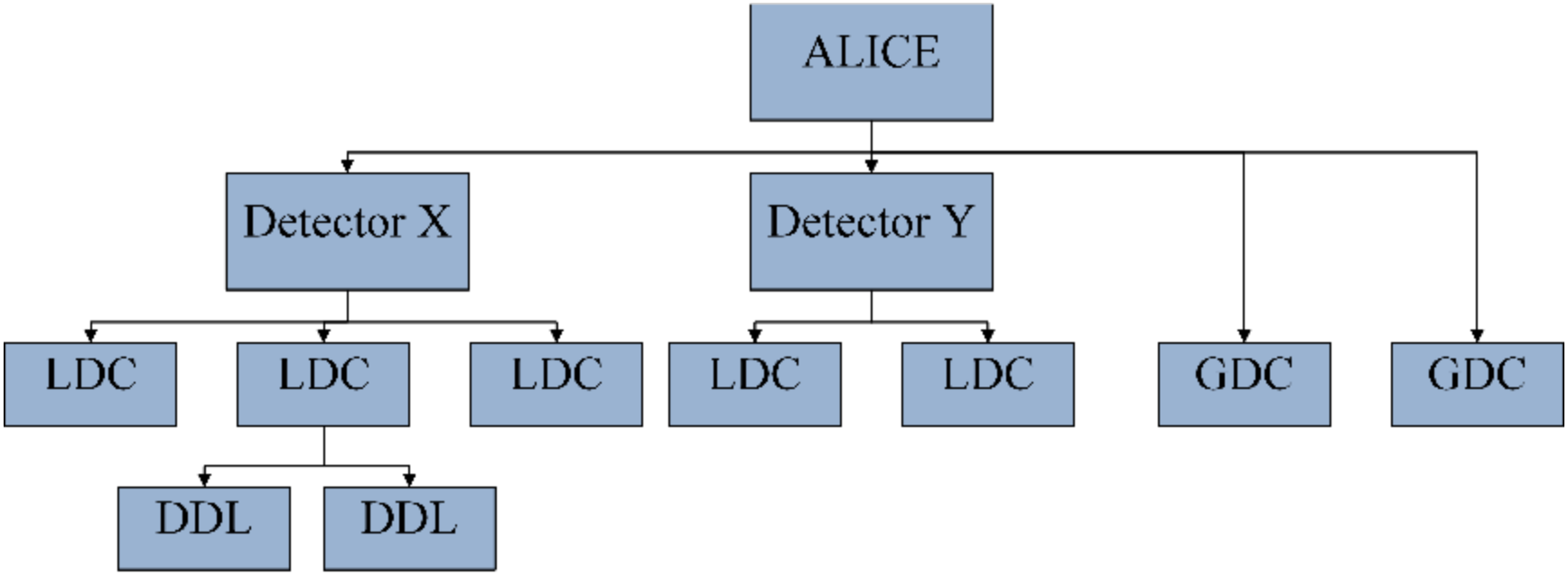
- A value in MonALISA is represented by the tuple (cluster, node, parameter, timestamp, value)
  - Abuse free-text fields cluster and node:



(1) DDL(X), LDC(Y), GDC(Z), DET(ABC), …

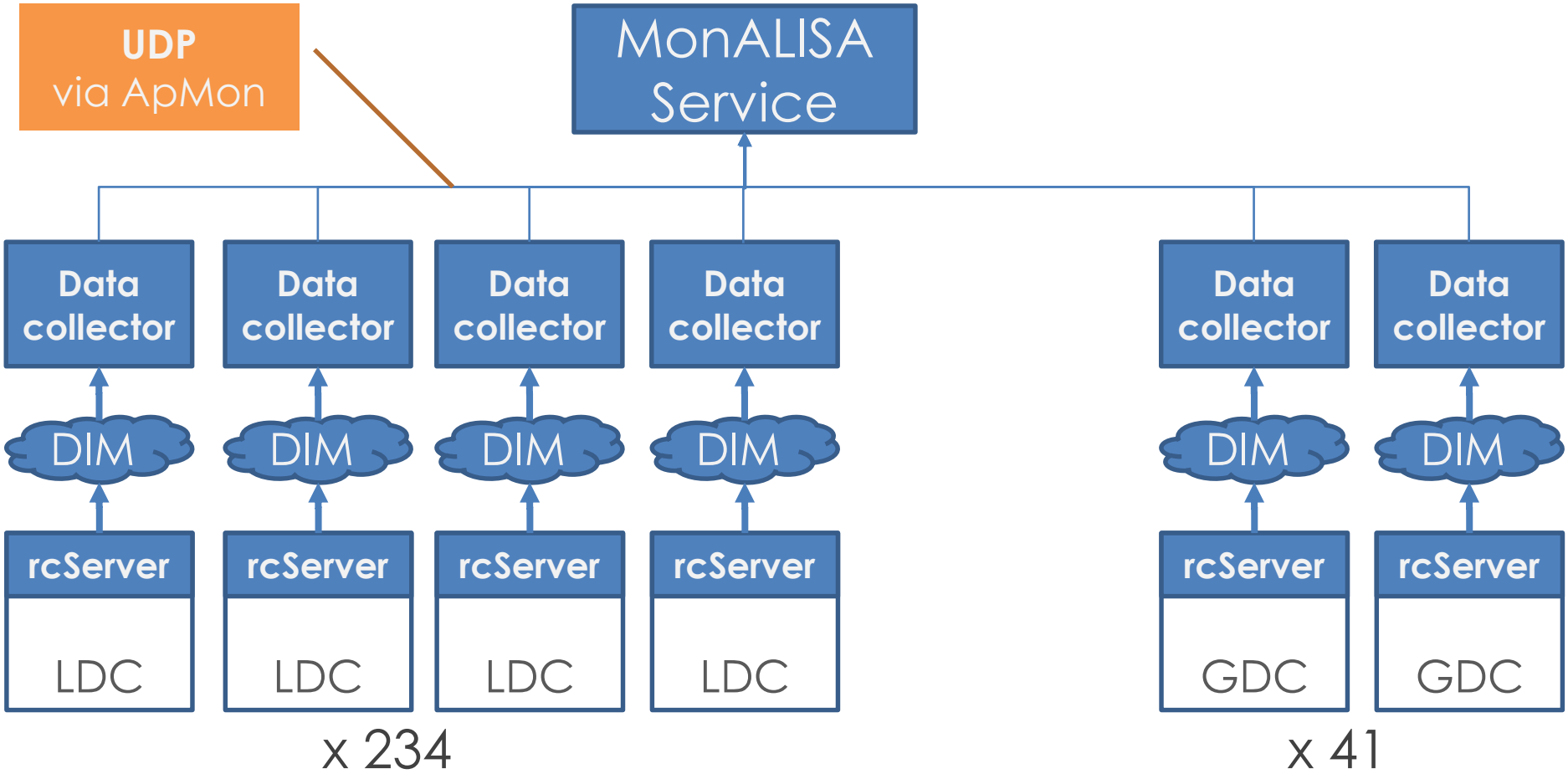
Data Model

- Implicit entities (node) hierarchy, follows ALICE control structure:



Data Collectors

- Data collectors are C++ DIM clients that push data via ApMon library to MonALISA Service every 5s



Data Filters

- Java classes running on MonALISA Service, subscribe to relevant values and perform aggregation into higher-level metrics

MonALISA Service

DDL
- Subscribes to link status
- Computes list of links with backpressure

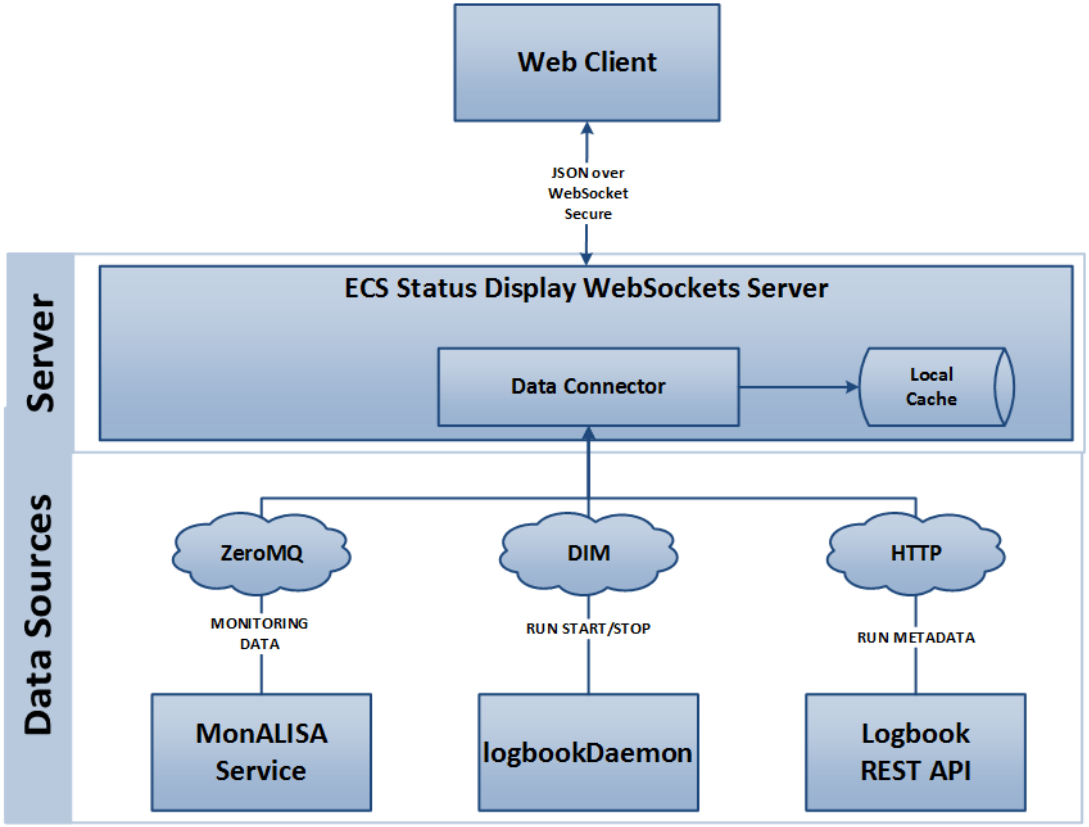
LDC
- Subscribes to HLT Pending Decisions, readout data rate, link status
- Computes MAX(HLT Pending Decisions), global readout rate, backpressure status per detector

GDC
- Subscribes to recording data rate, event rate, incomplete events
- Computes global recording rate, event rate (global/per trigger cluster), total incomplete events

ALICE
- Subscribes to all metrics to be displayed
- Sends them via ZeroMQ to Web Sockets Server

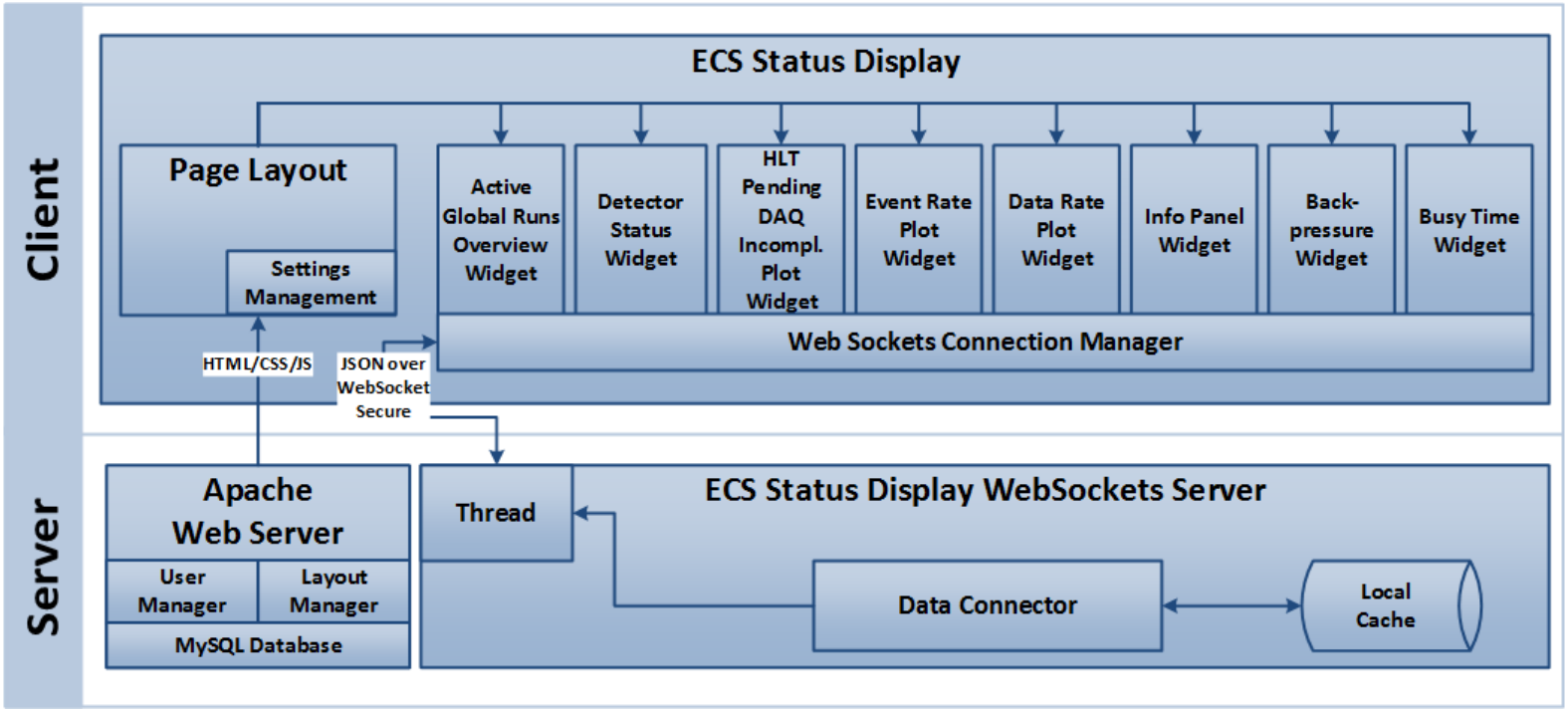
Web Sockets Server

- Collects data from multiple sources, keeps local cache, pushes values to clients
- Home-made PHP implementation of RFC 6455



Status Display

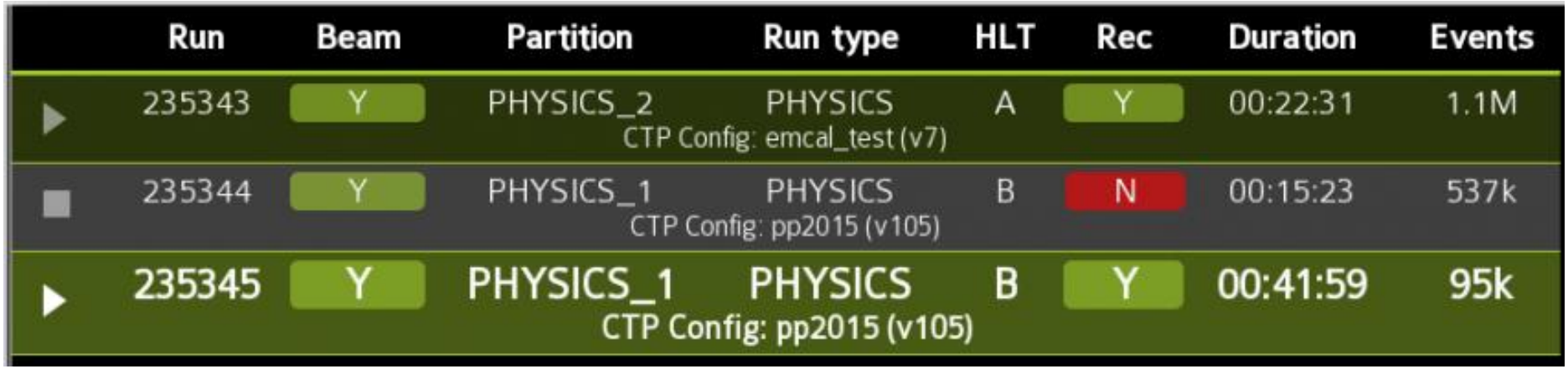
- Common layer handles connections, data
- Widgets subscribe to data channels
- Plots use *Flot* javascript library, have zoom window
- Replay runs, follow partition, auto switch, CERN SSO



Status Display

Status Display

- Global Overview
  - Active / Recent runs (up to 3)
  - Run #, Beam status, Partition, Run Type, HLT Mode, Recording, Duration, # of Events, Trigger Configuration
  - On run selection, other widgets are notified and can change display



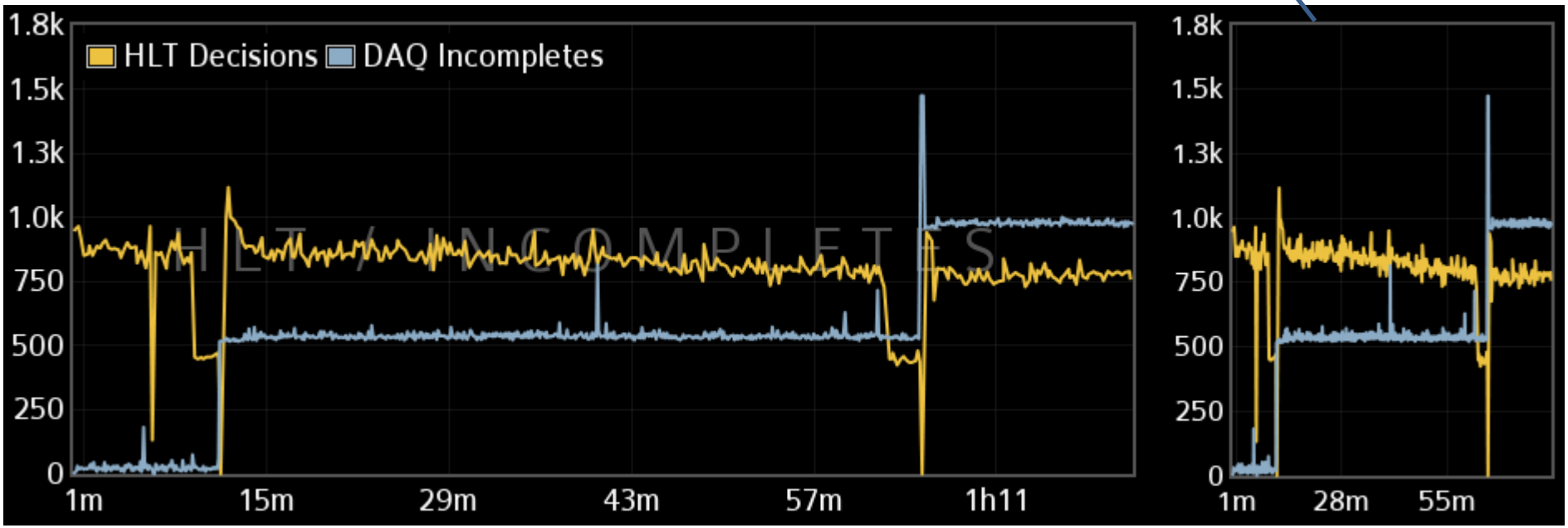
Status Display

- Detector Status
  - Participating detectors
  - Calibration runs of last 8 hours
  - Standalone runs
  - Backpressure
  - Busy
  - % of active links
  - Events per TC
  - Disabled trigger inputs

Status Display

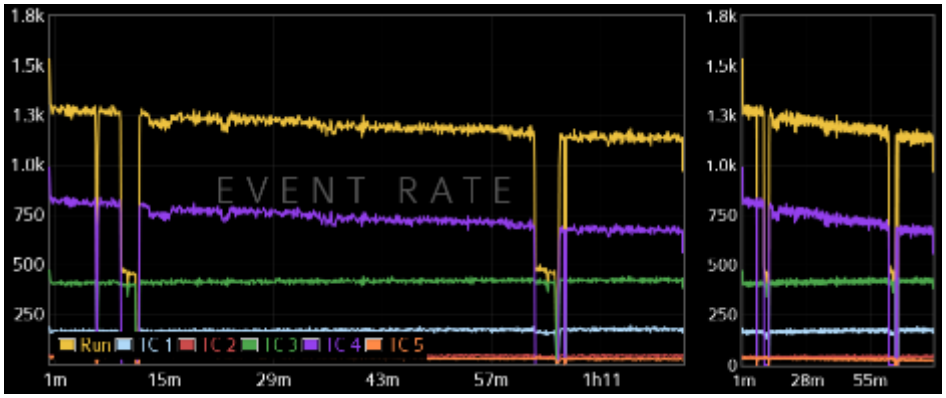
- HLT Pending Decisions
- DAQ Incomplete Events

In LIVE mode, latest 10 min

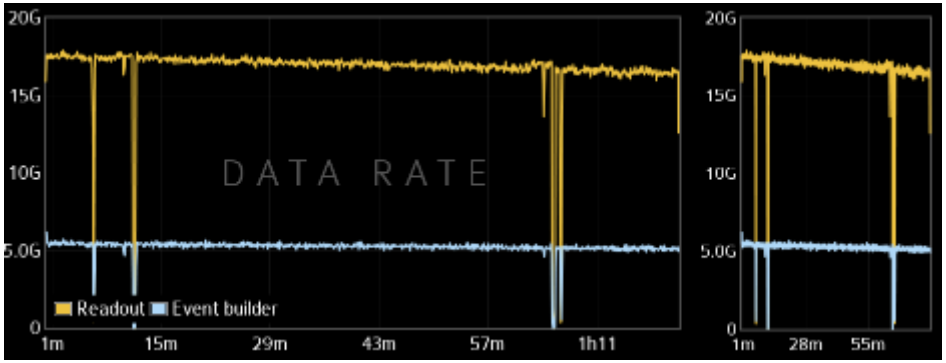


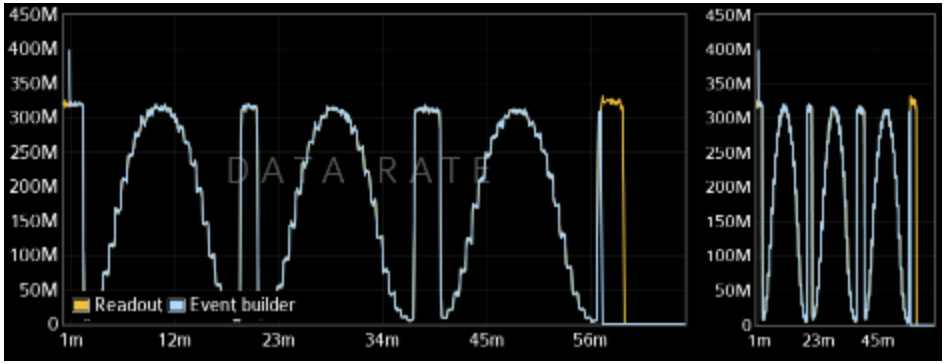
Status Display

- Event Rate (total and per Trigger Cluster)



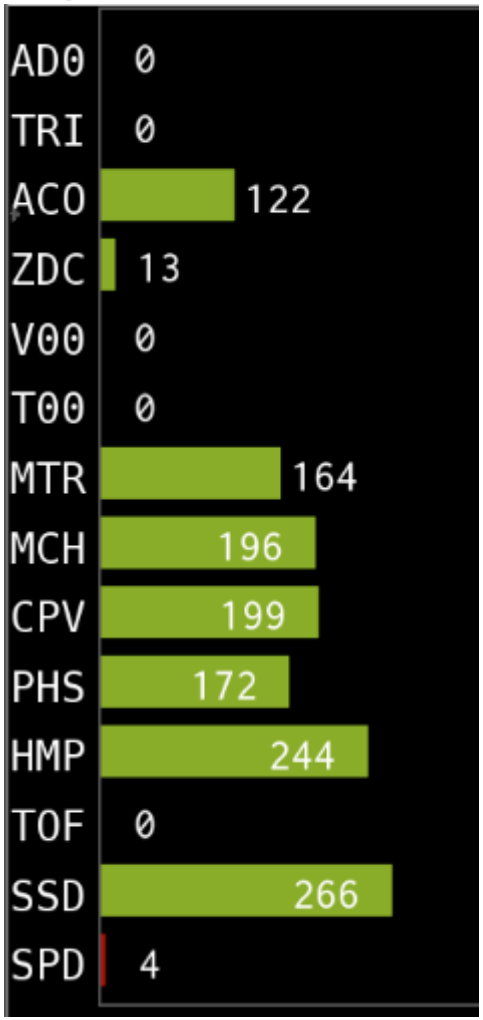
- Data rate (Readout and Event Builder)



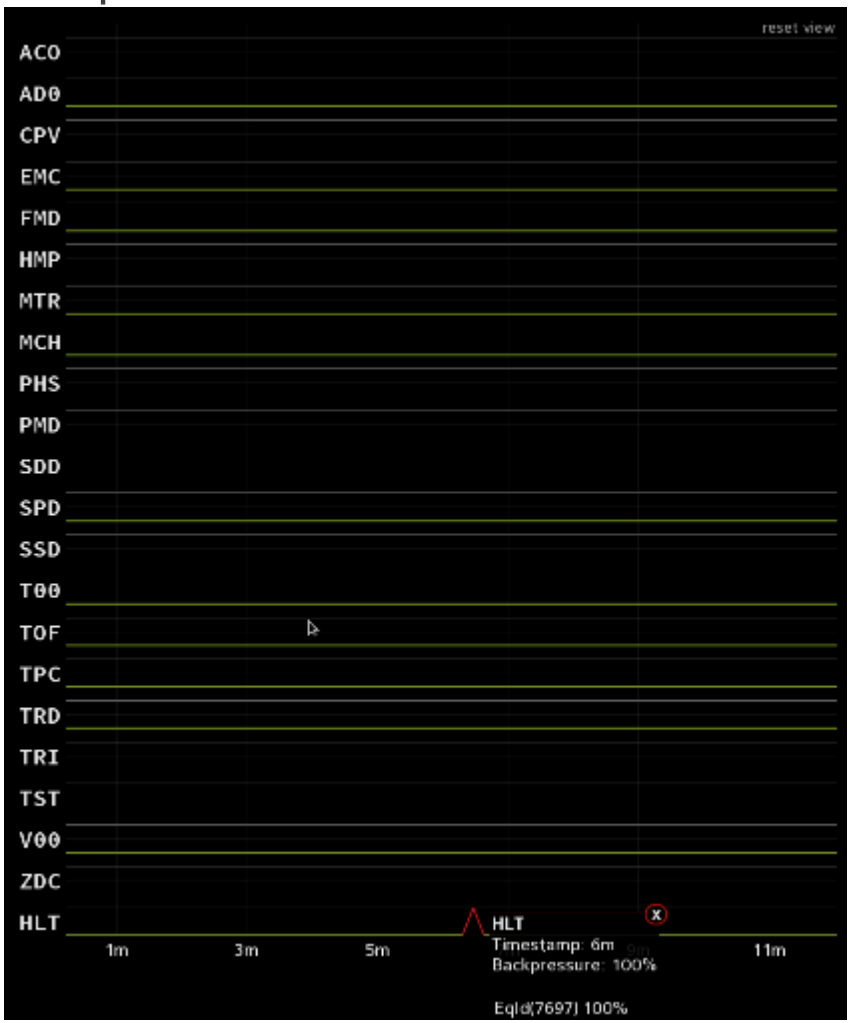


Status Display

- Busy time

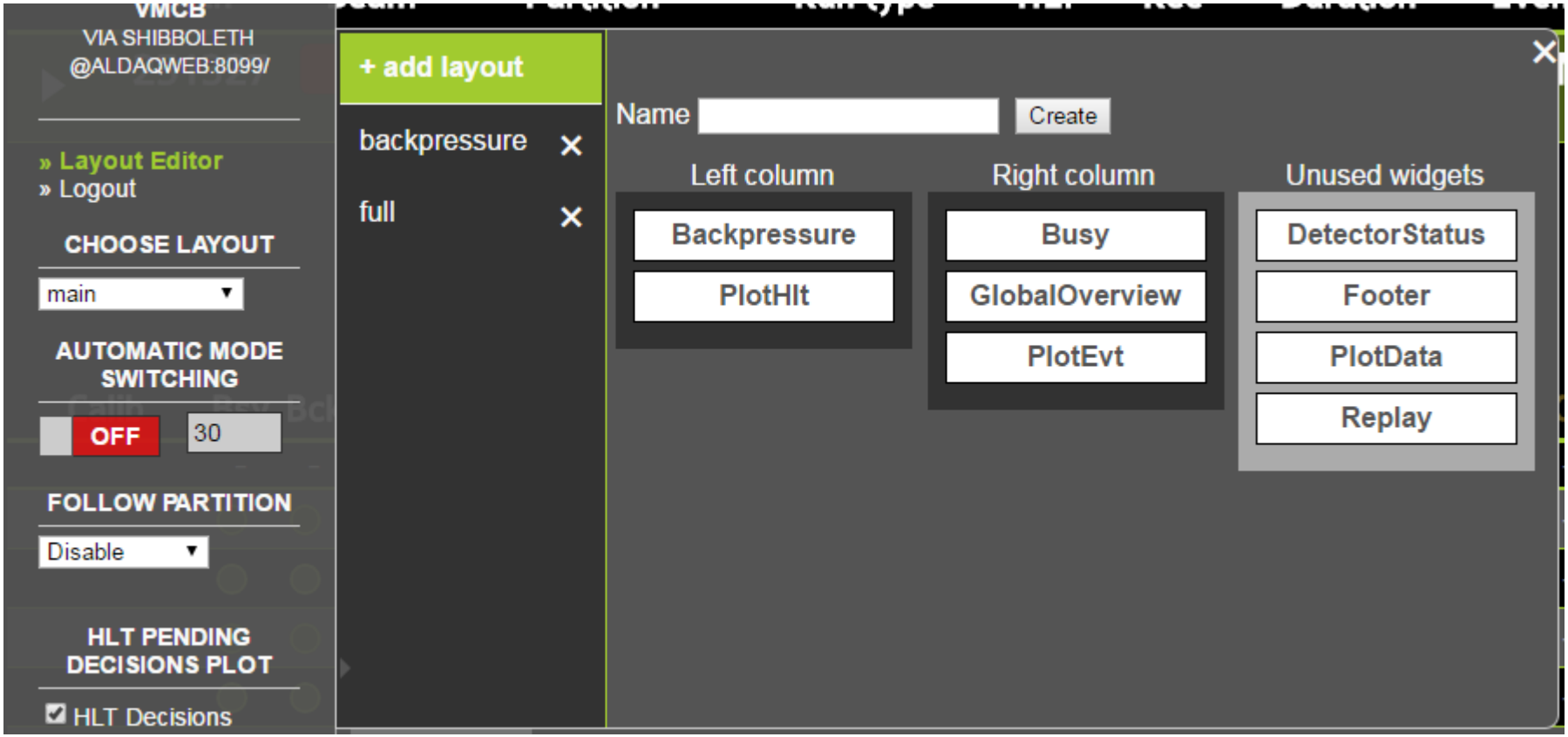


- Backpressure

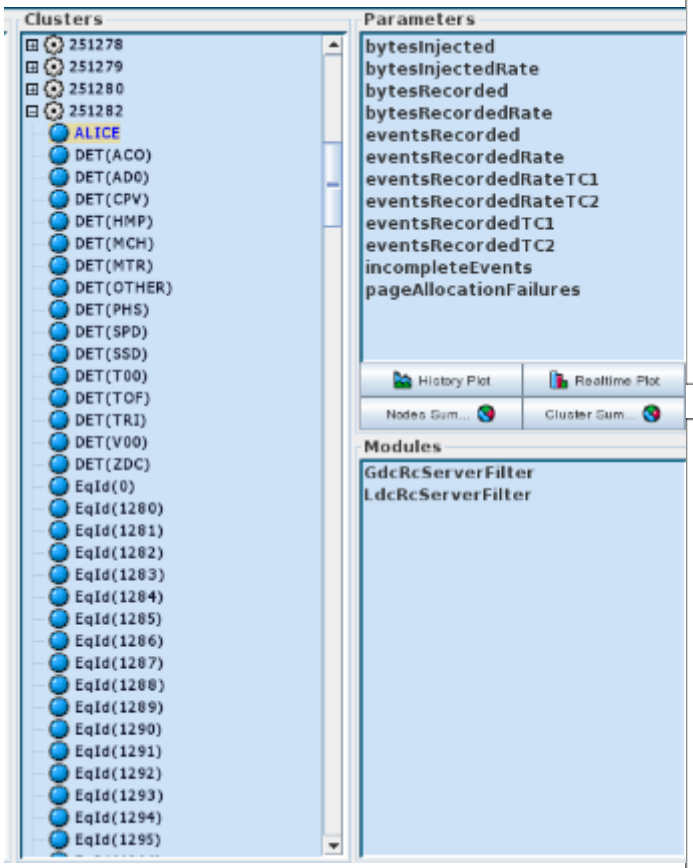


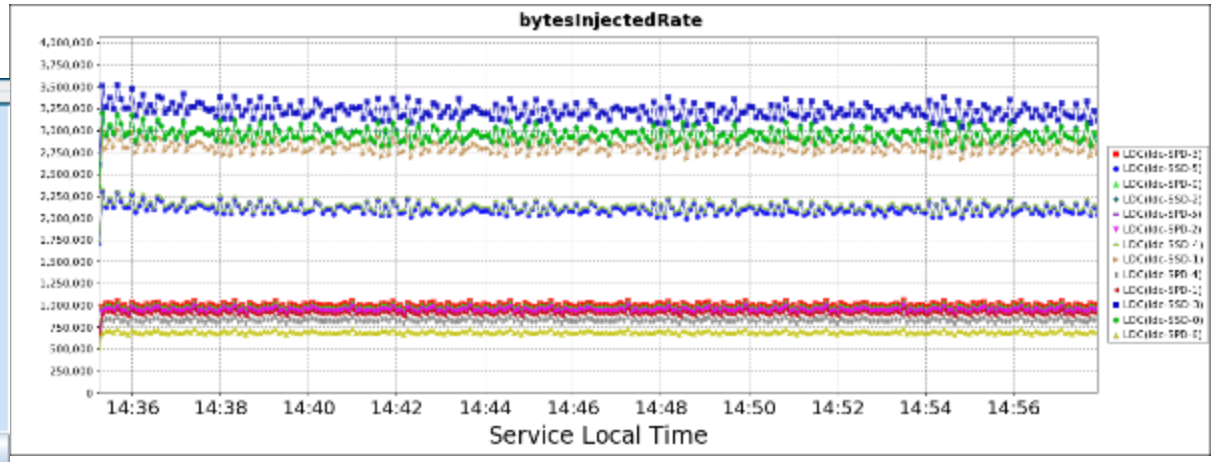
Status Display

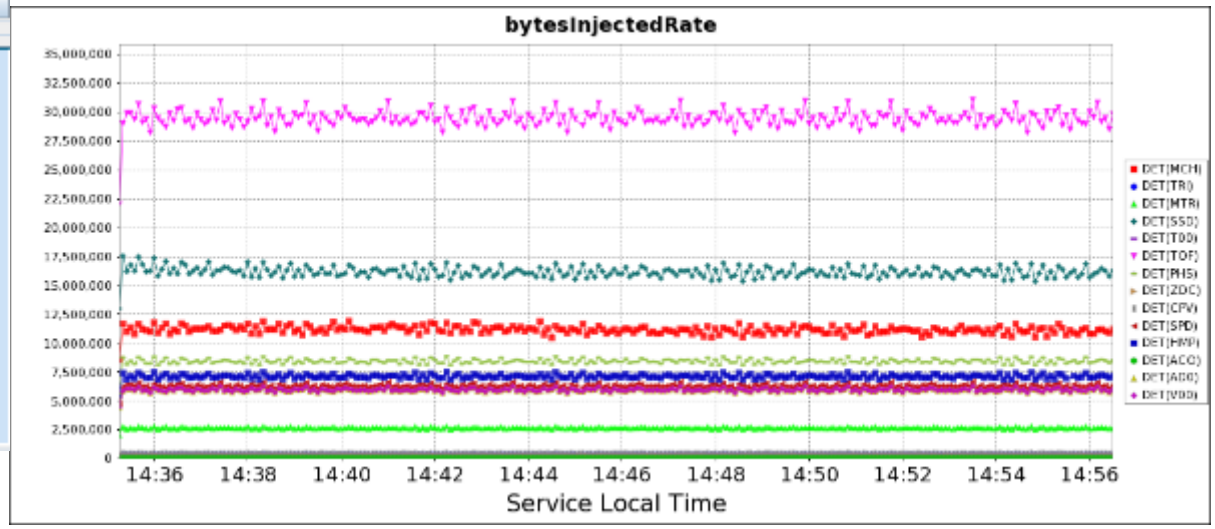
- Widgets can be organized by users in personalized layouts



MonALISA Client GUI







Future Work

- Include more metrics/widgets
  - Incomplete per detector, busy over time, buffer occupancy, …

Conclusion

- Dataflow Monitoring has been greatly improved in ALICE during LS1
- Extensively used by shift crew to detect abnormal situations
- Extensively used by experts to investigate abnormal situations
- Screenshots extensively used in presentations, reports, etc.
- Modular design allows for external teams to push data and develop widgets (e.g. busy time)