

TPC QA + experience with the AMORE  
framework

Marian Ivanov, Peter Christiansen  
+ GSI group

# *Outlook*

- ◆ TPC QA – What do we want ?
  - ◆ Raw data QA monitoring
  - ◆ Online monitoring of calibration parameters
- ◆ TPC QA Amore integration
- ◆ Experience and suggestions

# **TPC QA**

- The TPC (online) QA consist from two parts
  - Raw data monitoring e.g.
    - Amplitude spectra (1D, Profiles) ->require noise map + primitive calibration
    - Time dependence of mean Amplitude
  - Monitoring of calibration parameters
    - Check the detector behaviour
    - Check the calibration algorithm itself
    - The output of the calibration algorithm to be used in reconstruction, desirable no calibration reiteration needed

# ***Quality assurance***

- Process:
  - Detect the problems - Define, what is the problem
  - What do we expect?
    - Defined in the TDR and in the PPR on the basis of simulation
  - Until which point the detector the information form the detector is reasonable?
    - How far we are from the expectation?
  - Define the limits of working conditions
  - Modify expectation
- TPC Strategy – Enable Expert mode of the QA just from the beginning.
  - Default histograms – views – configurable - generated by expert monitor
- More details about TPC QA -  
<https://alisoft.cern.ch/AliRoot/trunk/TPC/doc/calib/calibrationTPC.pdf>

# Raw data monitoring

## Peter Christiansen

# *TPC Online Monitoring*

- ◆ “Old” online monitors (/ RAW data monitors)
  - ◆ Expert pad monitor
  - ◆ AliEVE event display
  - ◆ Single events
  - ◆ No cluster information: only “digits”
- ◆ ALICE QA framework supports online monitoring via AMORE
  - ◆ New QA RAW monitor based on GSI TPC calibration framework with simple clustering
  - ◆ Integrates with GUI viewer
    - ◆ Also GUI viewer is now working in AMORE
  - ◆ Average cluster characteristics over many events

# *TPC QA structure*

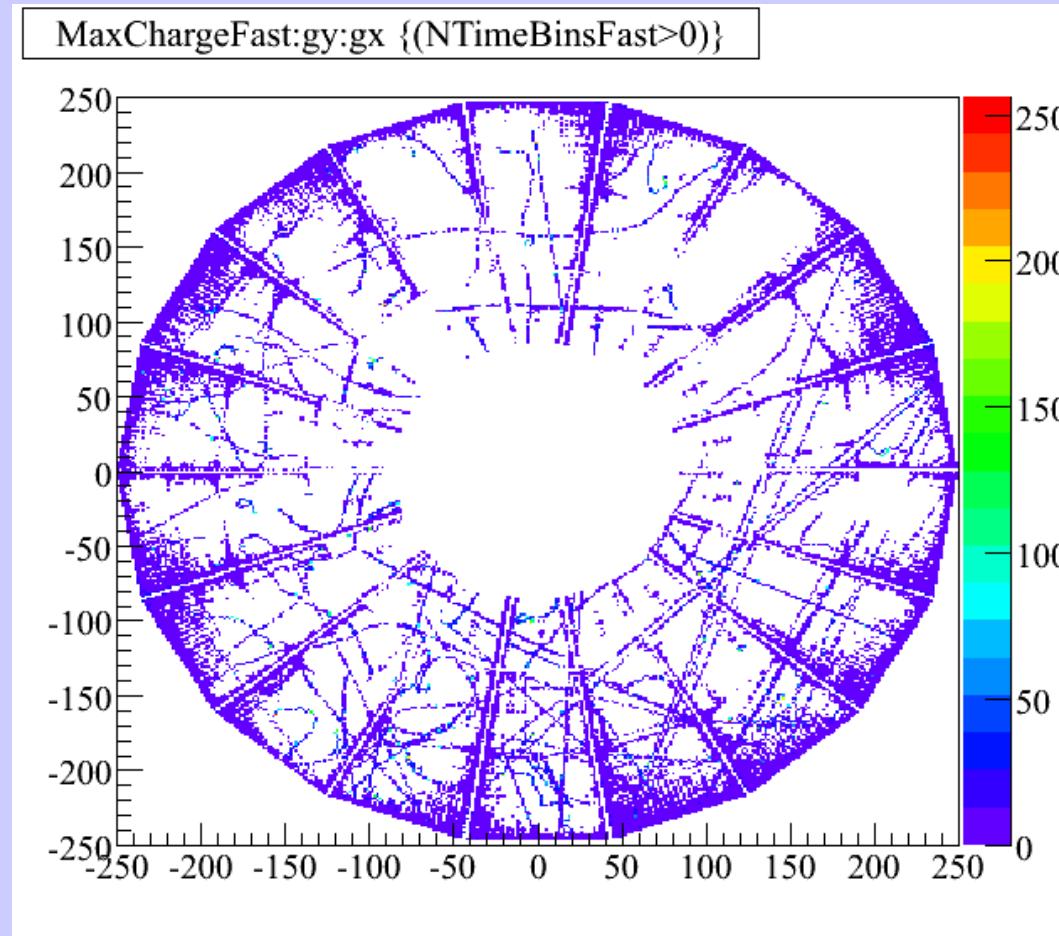
- ◆ Based on PHOS classes
- ◆ In AliRoot since January 2008
- ◆ Simulated data (AliTPCQADataMakerSim)
  - ◆ QA for Hits and Digits (Sdigits not supported)
  - ◆ Main focus on Hits (important for validation of simulation models, e.g., FLUKA (& GEANT4?))
- ◆ Reconstructed data (AliTPCQADataMakerRec)
  - ◆ QA for Raw data, RecPoints and ESDs
  - ◆ Main focus on RAW data (to be used for online monitoring)
  - ◆ The actual RAW data QA is done in the class AliTPCdataQA (based on the TPC calibration class)

# *The RAW data QA*

*Updated February/March 2008*

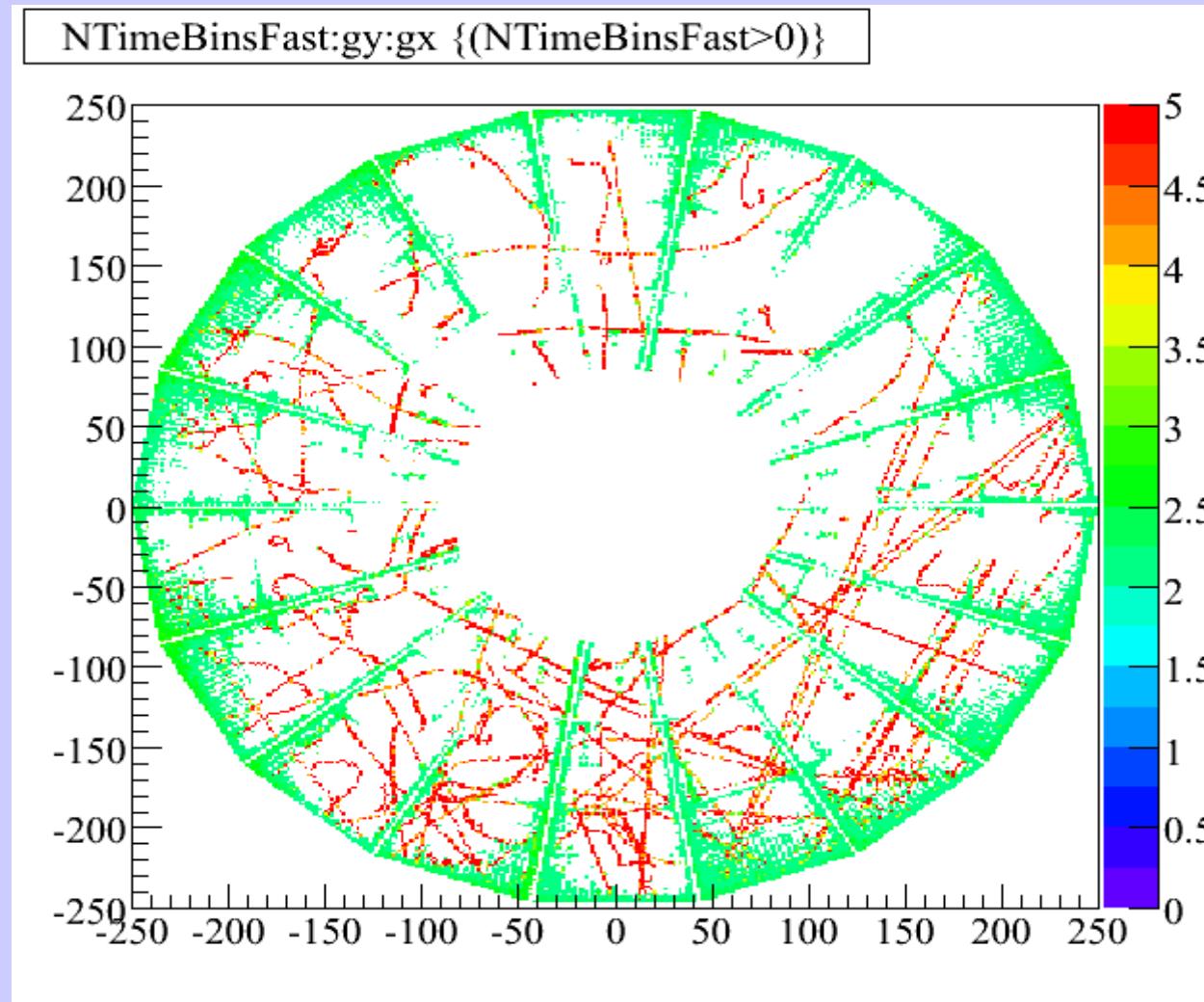
- ◆ Simple clustering implemented to have access to cluster characteristics
  - ◆ Raw data are expanded sector by sector in memory
  - ◆ Local extrema are identified
    - ◆ This is a simplified version of the clustering implemented in AliTPCclustererMI
  - ◆ We can then study e.g. the gain ( $Q_{\max}$ ,  $Q_{\text{tot}}$ )
- ◆ Integrated into AMORE online monitoring framework (Filimon and Marian)
  - ◆ Old version tested with online data
  - ◆ New version tested with files on monitoring machine

# *RAW data QA: Cosmic data from December run*

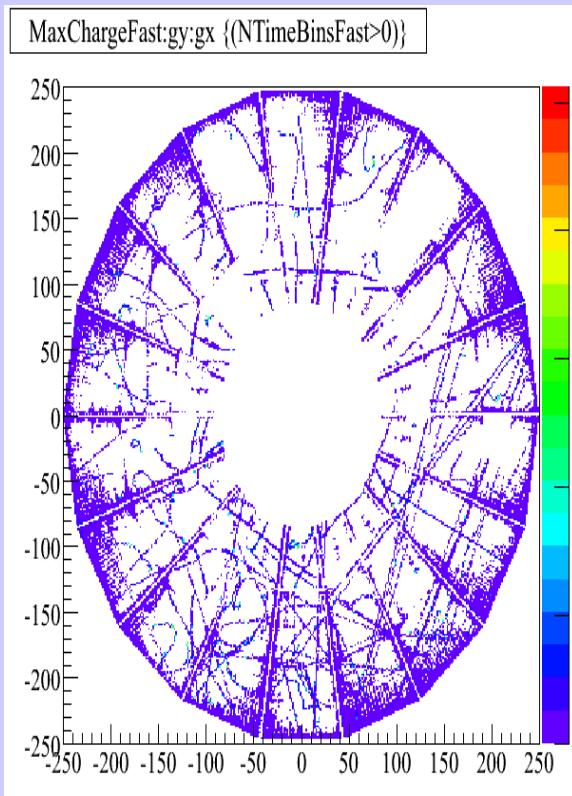


- ◆ Output can be viewed (and manipulated!) in TPC calibration browser online

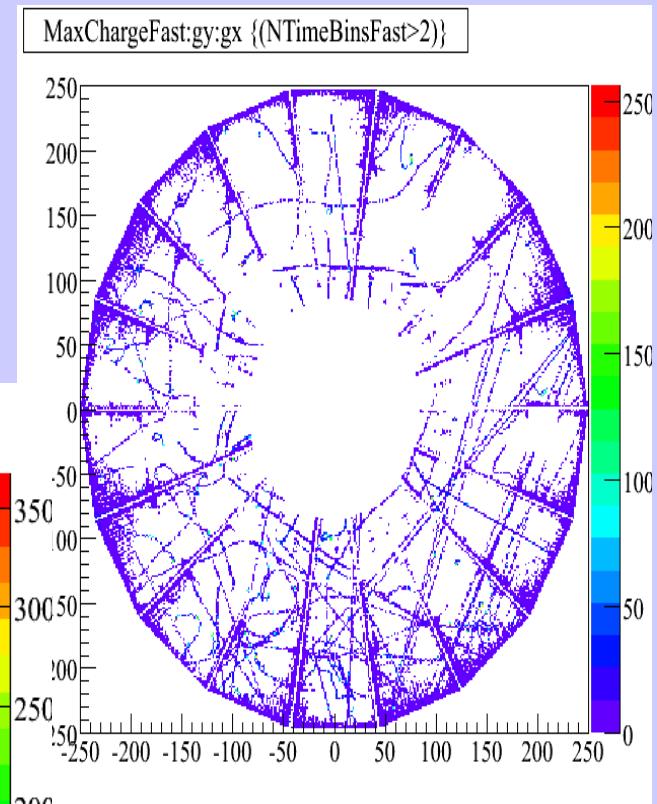
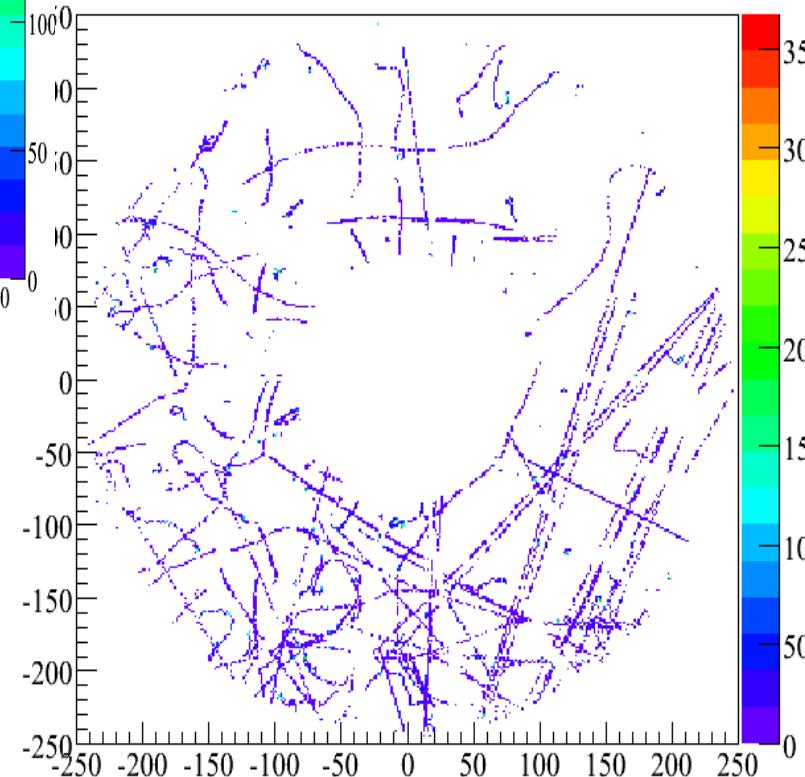
*The noise (not shaped!) can be removed since also the mean number of timebins is available*



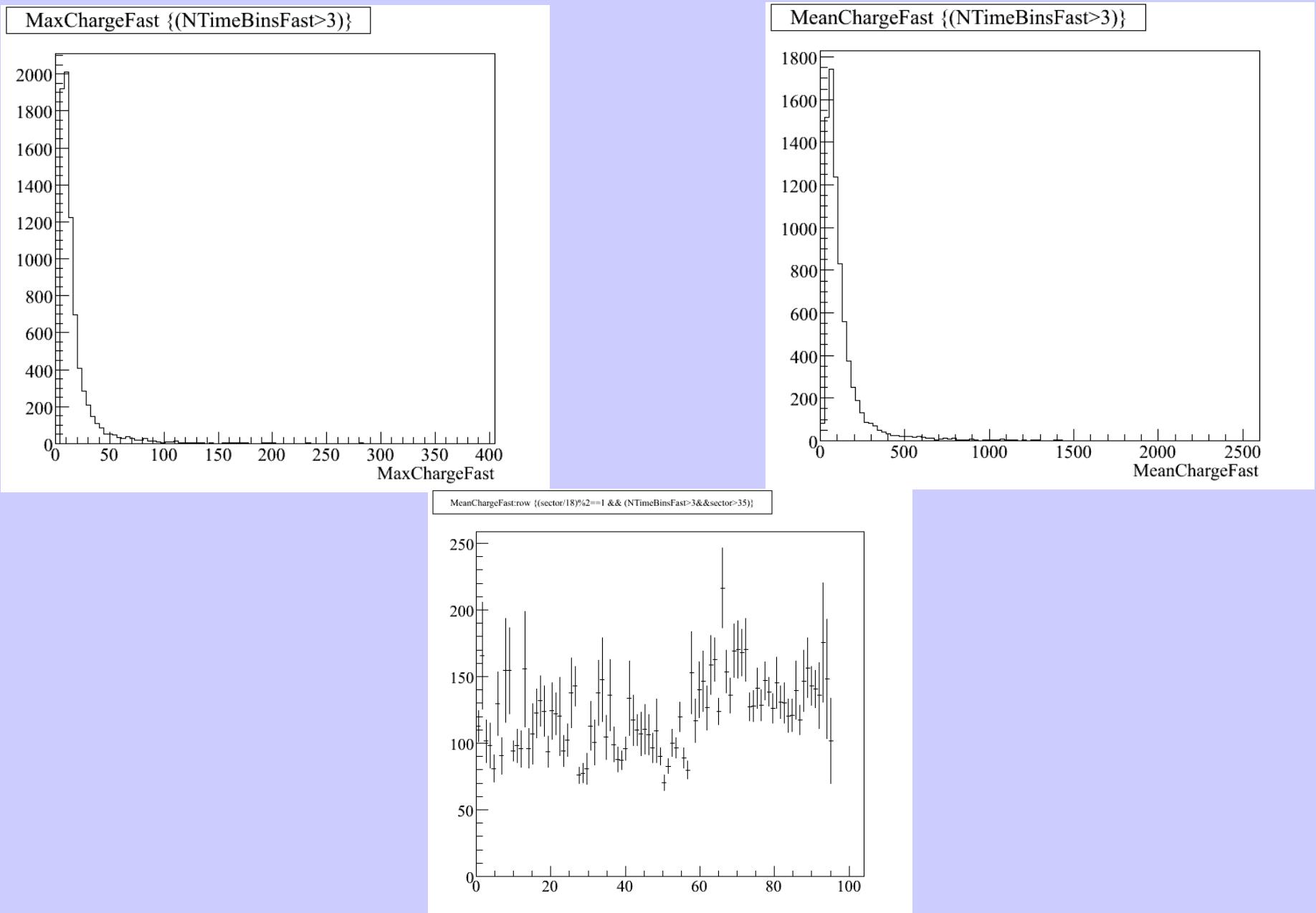
# *Qmax with cuts on mean time bins*



axChargeFast:gy:gx {(NTimeBinsFast>3)}



# *Q and Qmax plots with cuts*



# *Current information*

## *(persistent for online mode)*

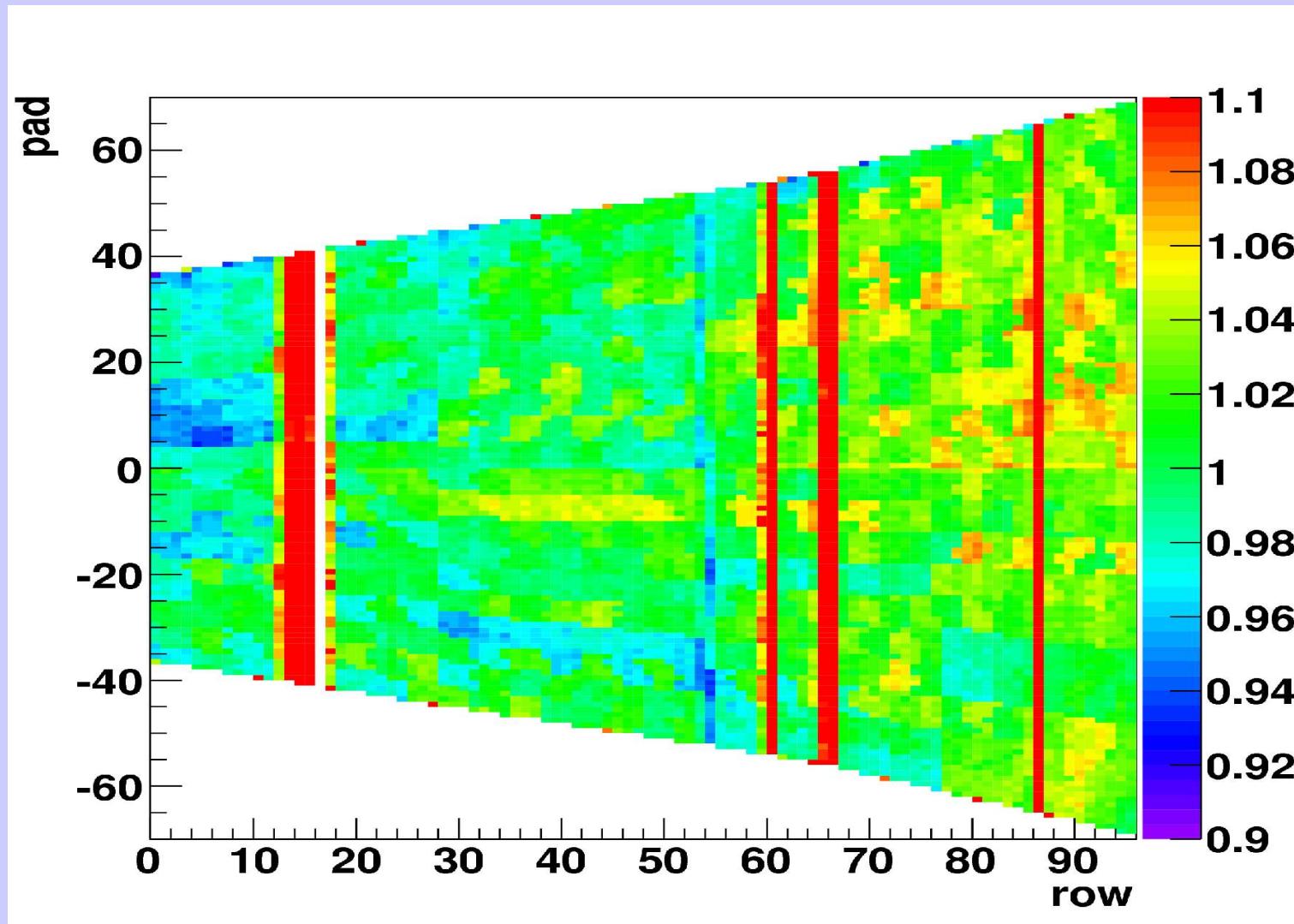
- ◆ NB! this is an average information for each pad. The monitor is only 2d (pad+row), so there is NO time dependence!
- ◆ **MaxCharge** –  $Q_{\max}$
- ◆ **MeanCharge** –  $Q_{\text{tot}}$
- ◆ **NlocalMaxima** – number of clusters
- ◆ **NPads** – pad width of cluster
- ◆ **NTimebins** – time width of cluster
- ◆ **NoThreshold** – fraction of digits above threshold (ZS reduction)
- ◆ **TimePosition** – mean Z position of clusters
- ◆ Other variables can be added, BUT there is an online memory limit on the number of variables so 1 new variable means that 1 old has to disappear

# Calibration (DA) QA

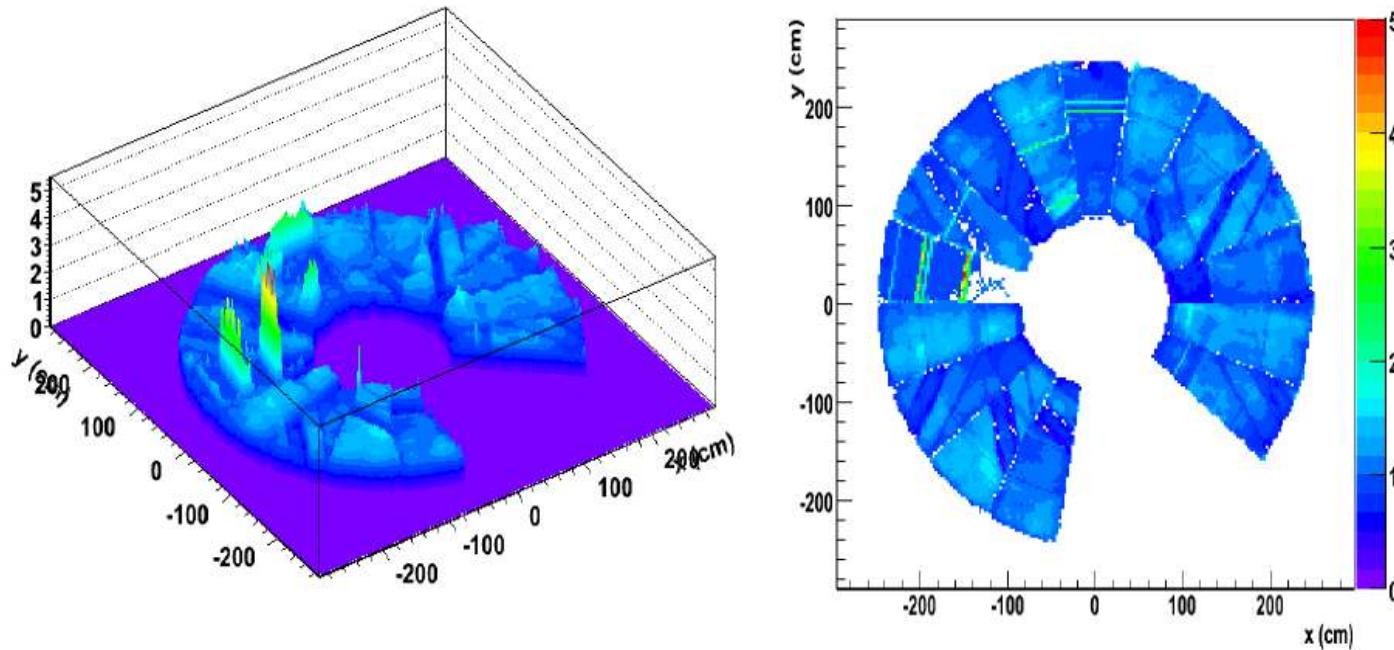
# ***Quality assurance (DA)***

- Data quality monitoring based on statistical properties of data - Extracting in calibration procedure
- Low level – digit level:
  - Noise and pedestal calibration
  - Electronic gain calibration
  - Time 0 calibration
- Direct answer
  - Number of dead channels
  - Percentage of suspicious channels
- Alarms:
  - ? To be defined ? - Consult with Hardware people

# *Pulser Q measurement*

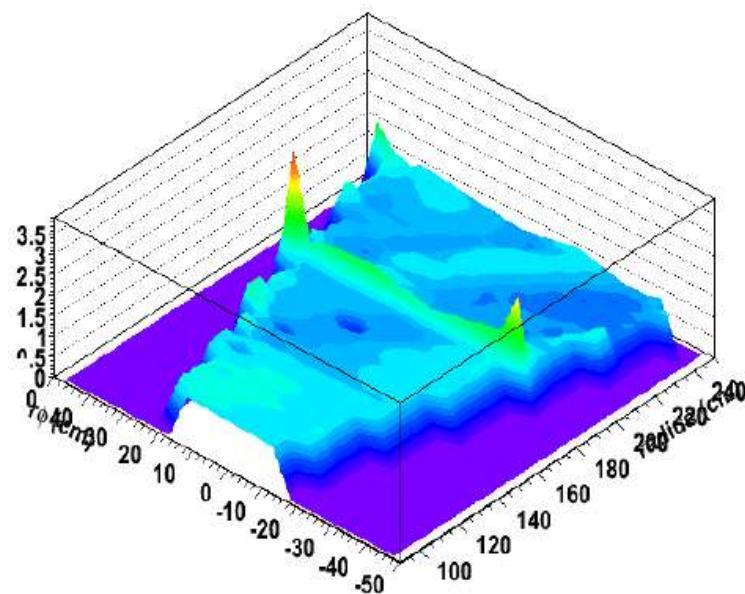
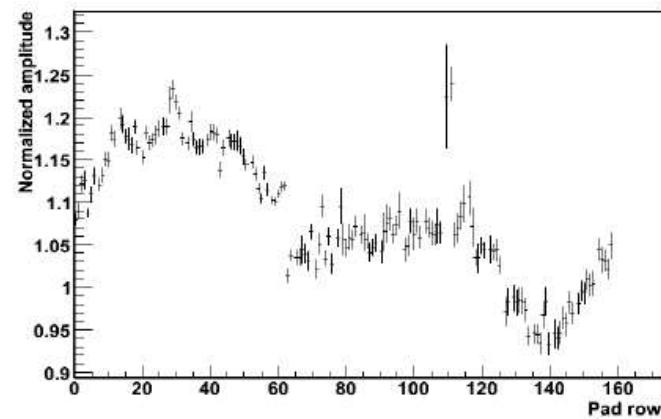
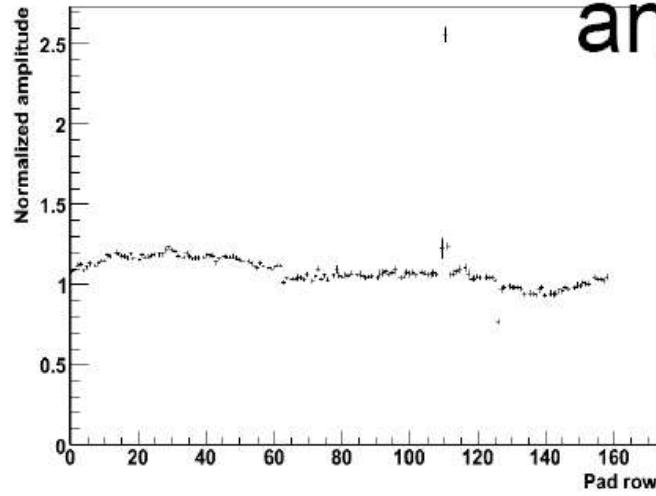


# Central electrode amplitude analysis



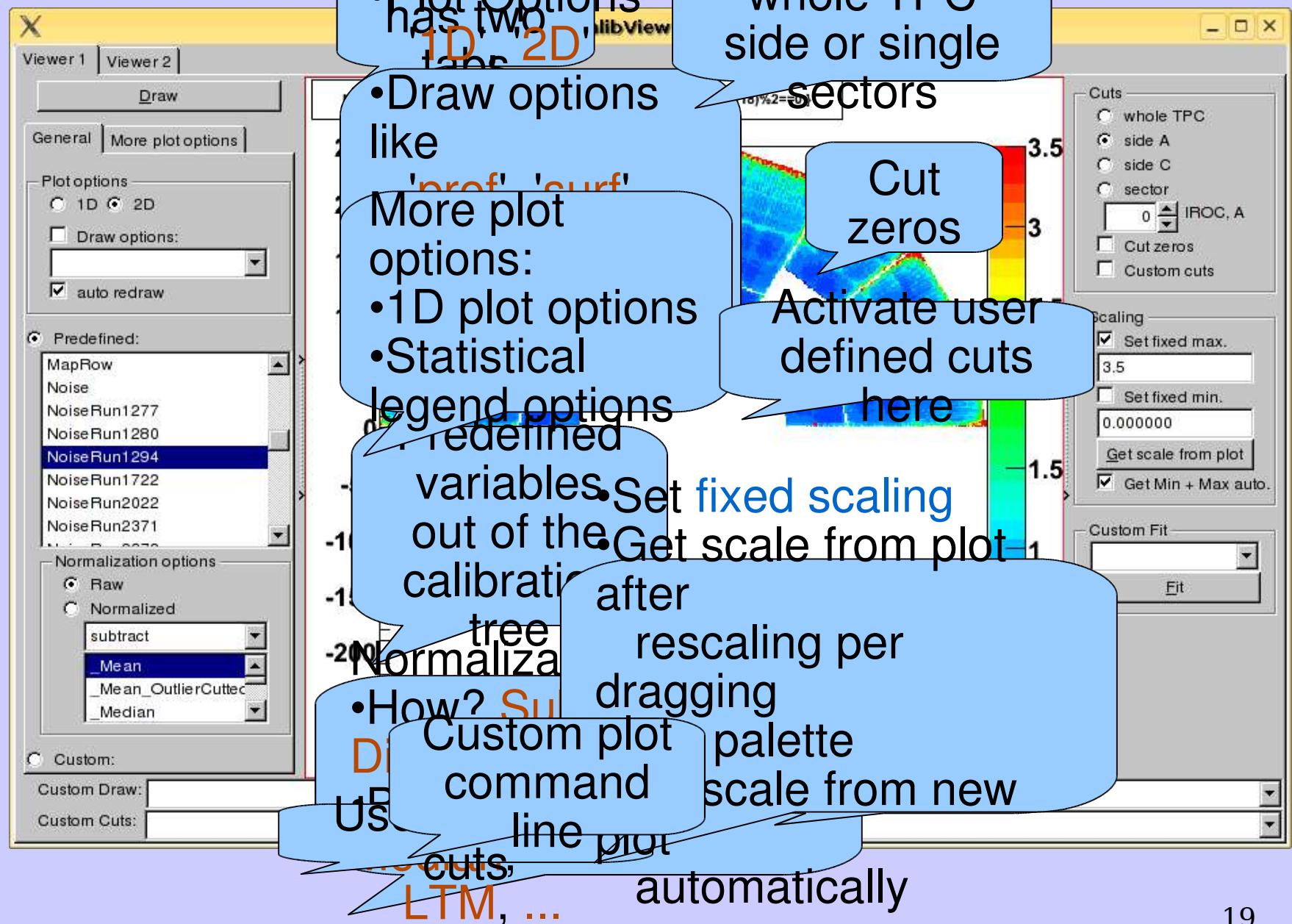
Marian Ivanov, Jens Wiechula

# Central electrode amplitude analysis

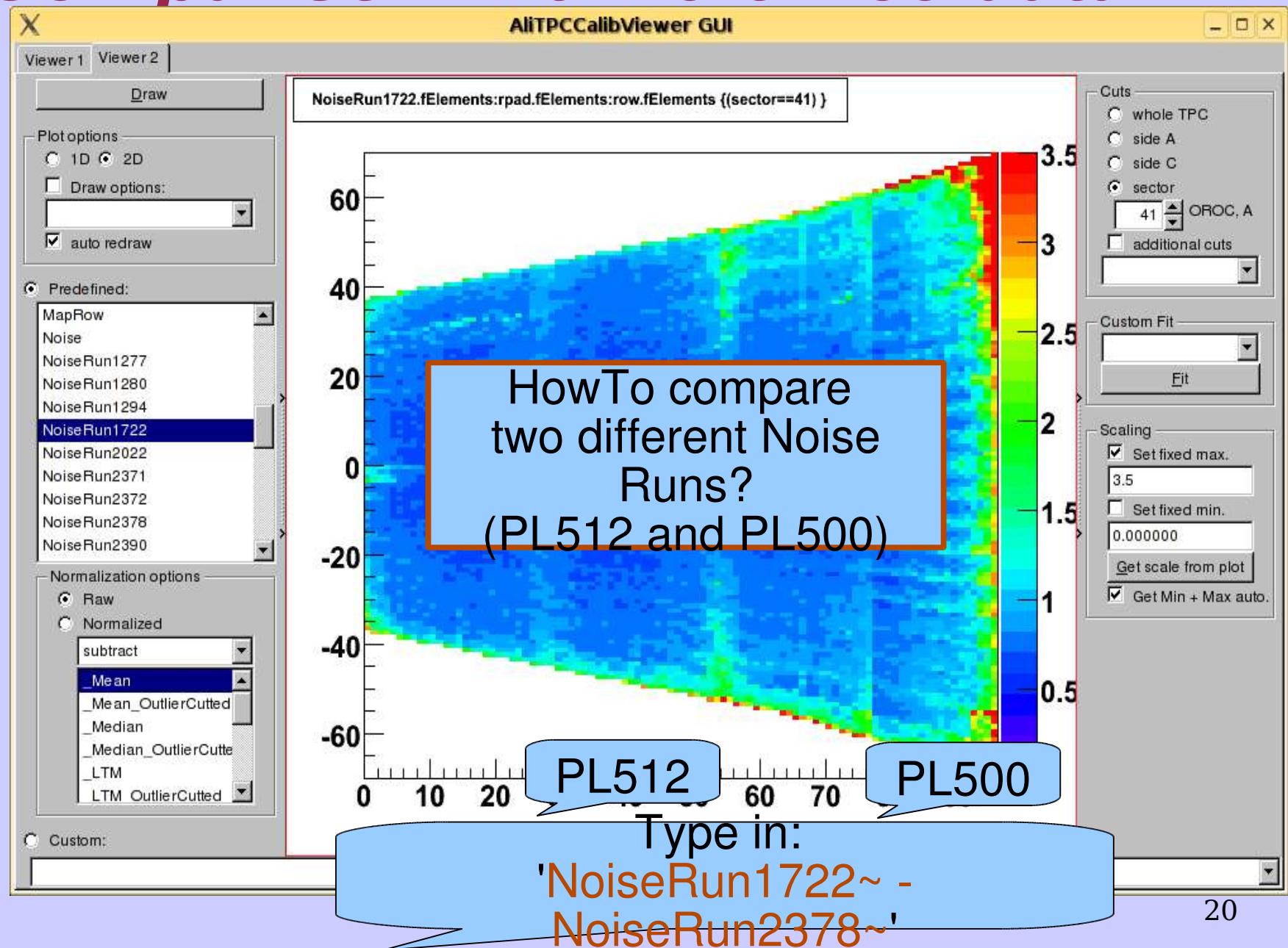


lens Wiechula

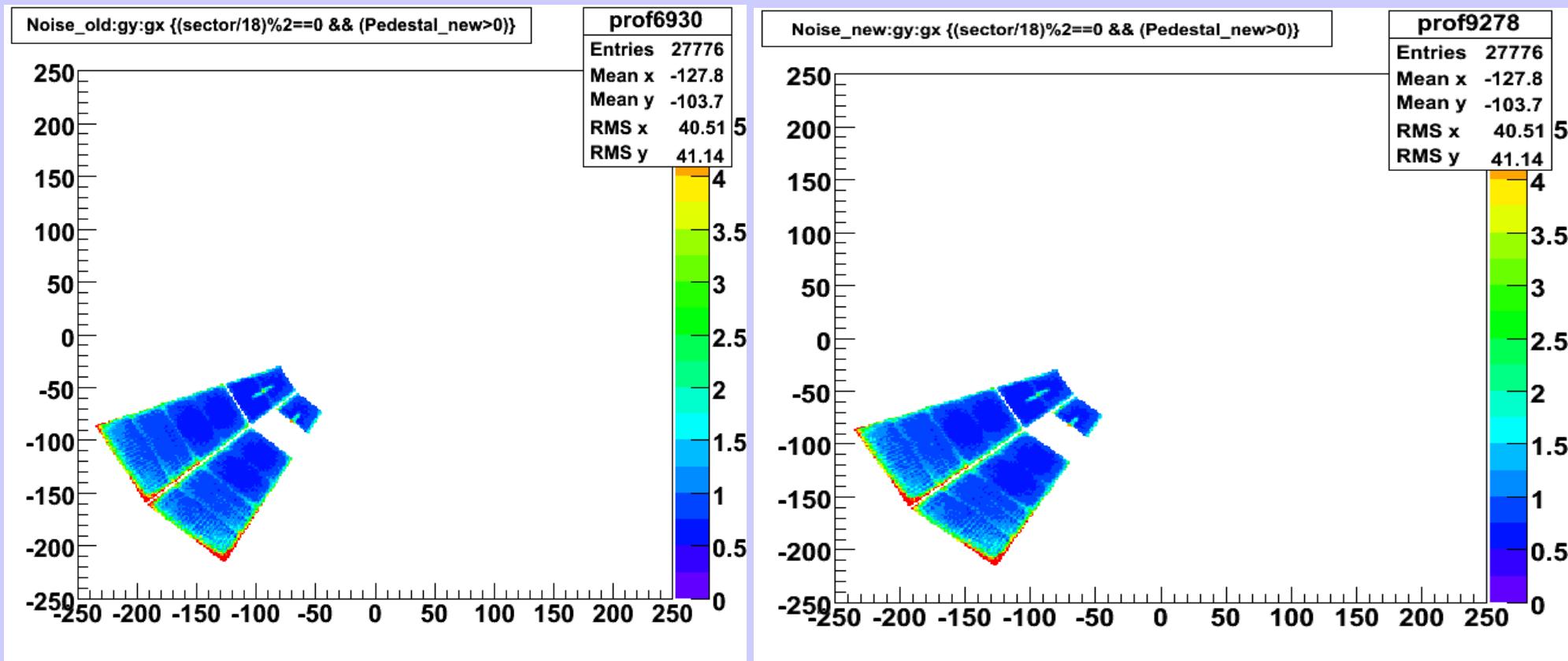
# GUI in detail



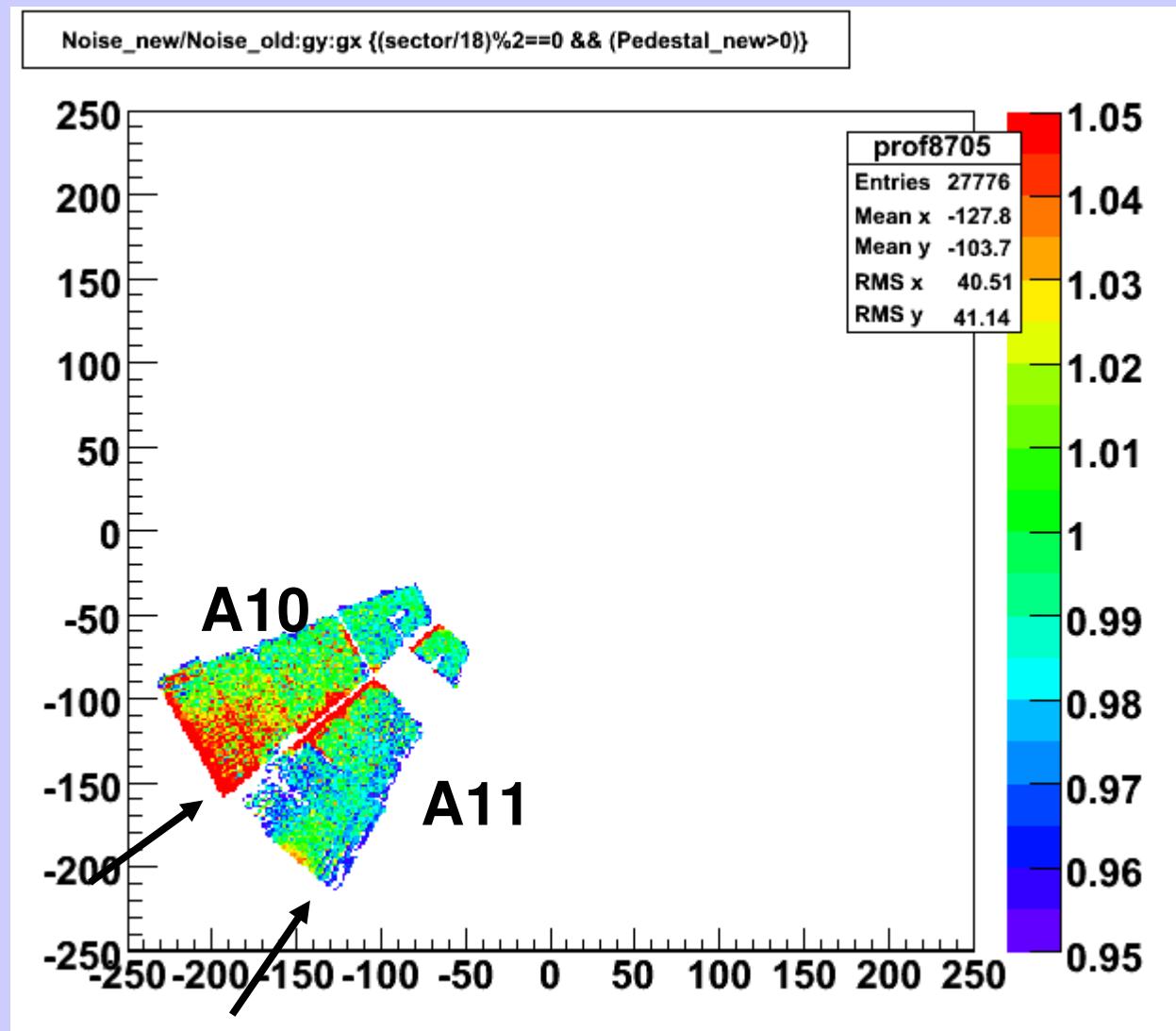
# *GUI demonstration – Comparison with reference data*



# Noise reference and new measured noise comparison

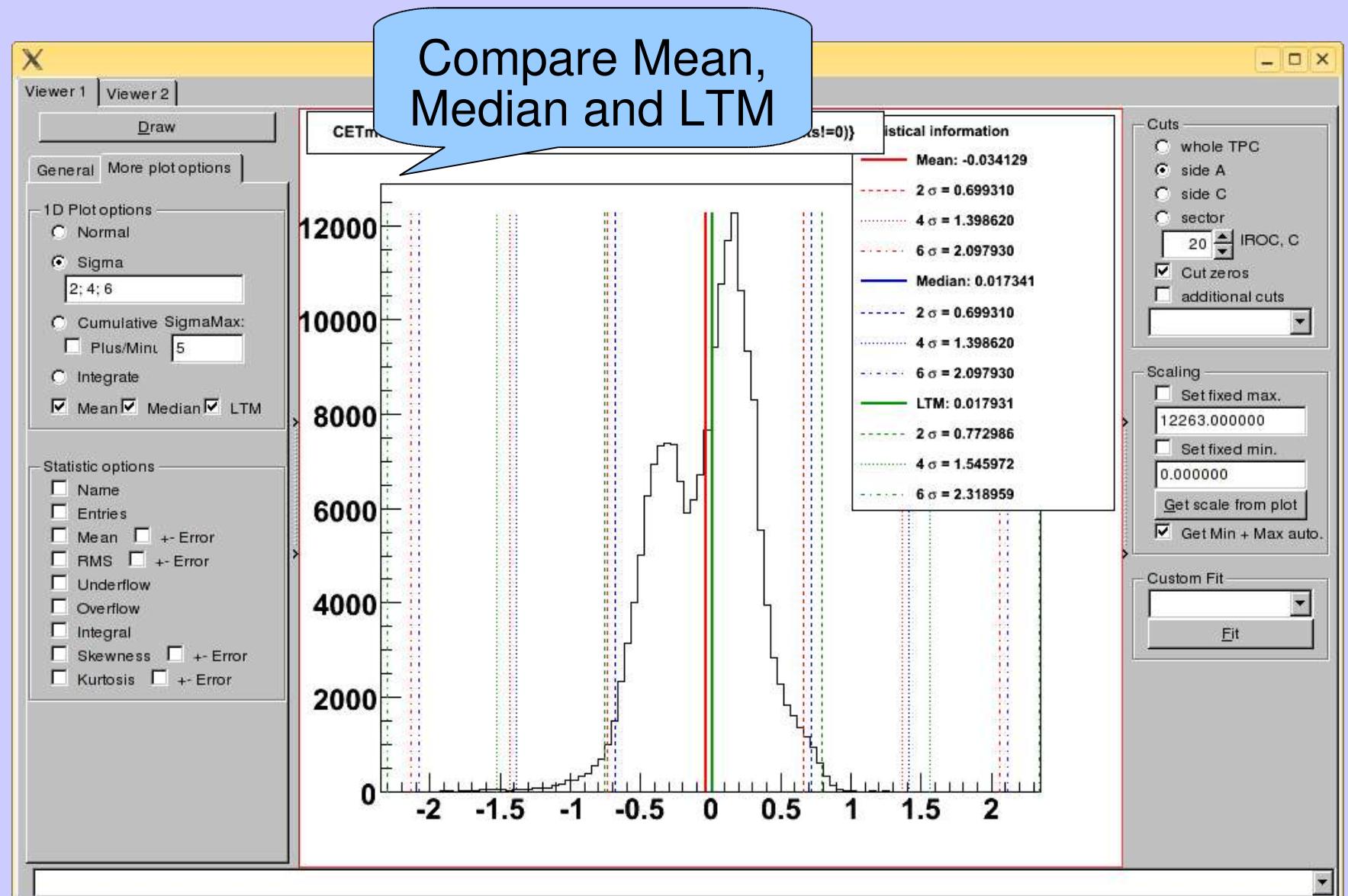


# Noise ratio new/old

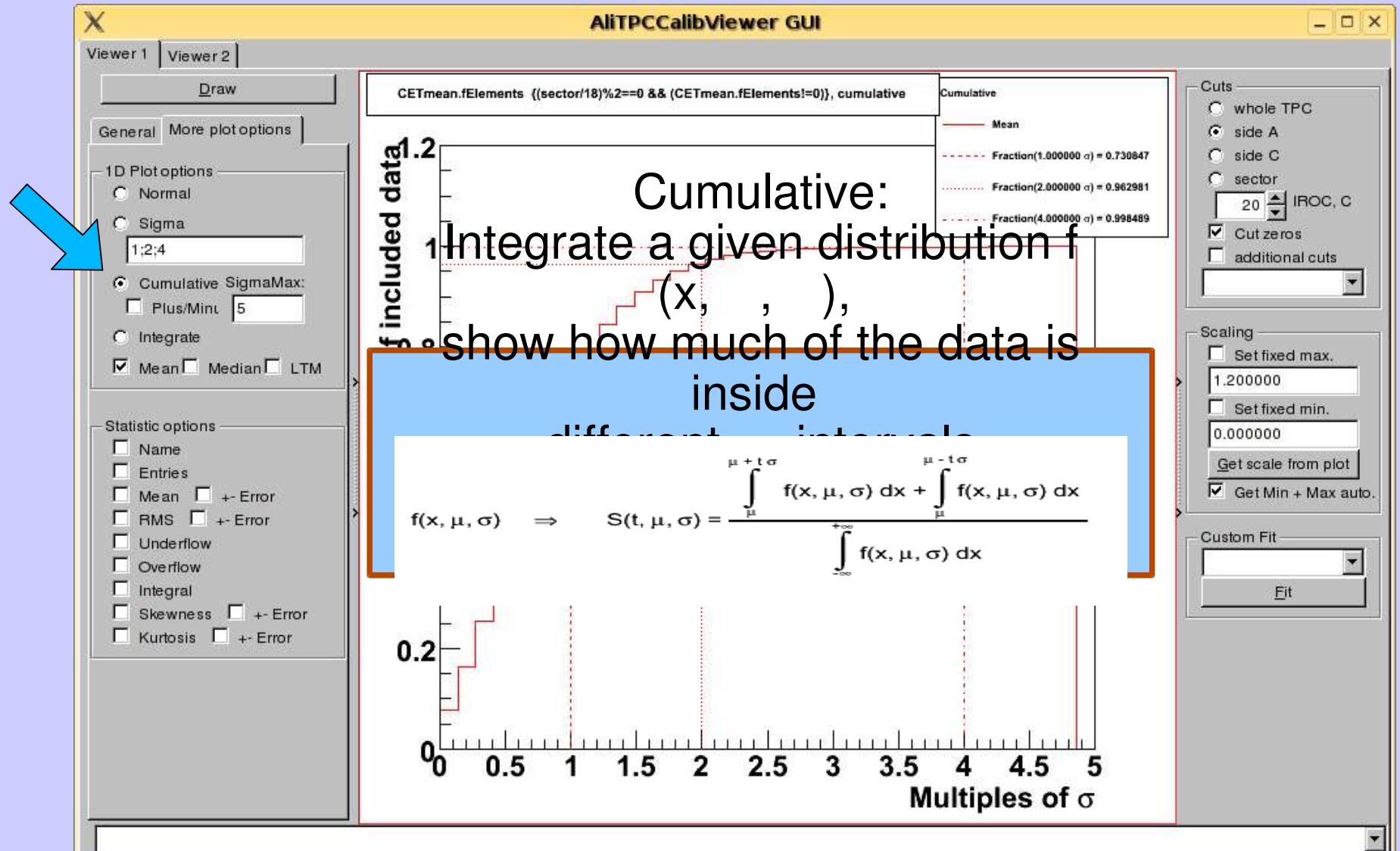


I have marked the busbars with arrows based on Rainers drawings

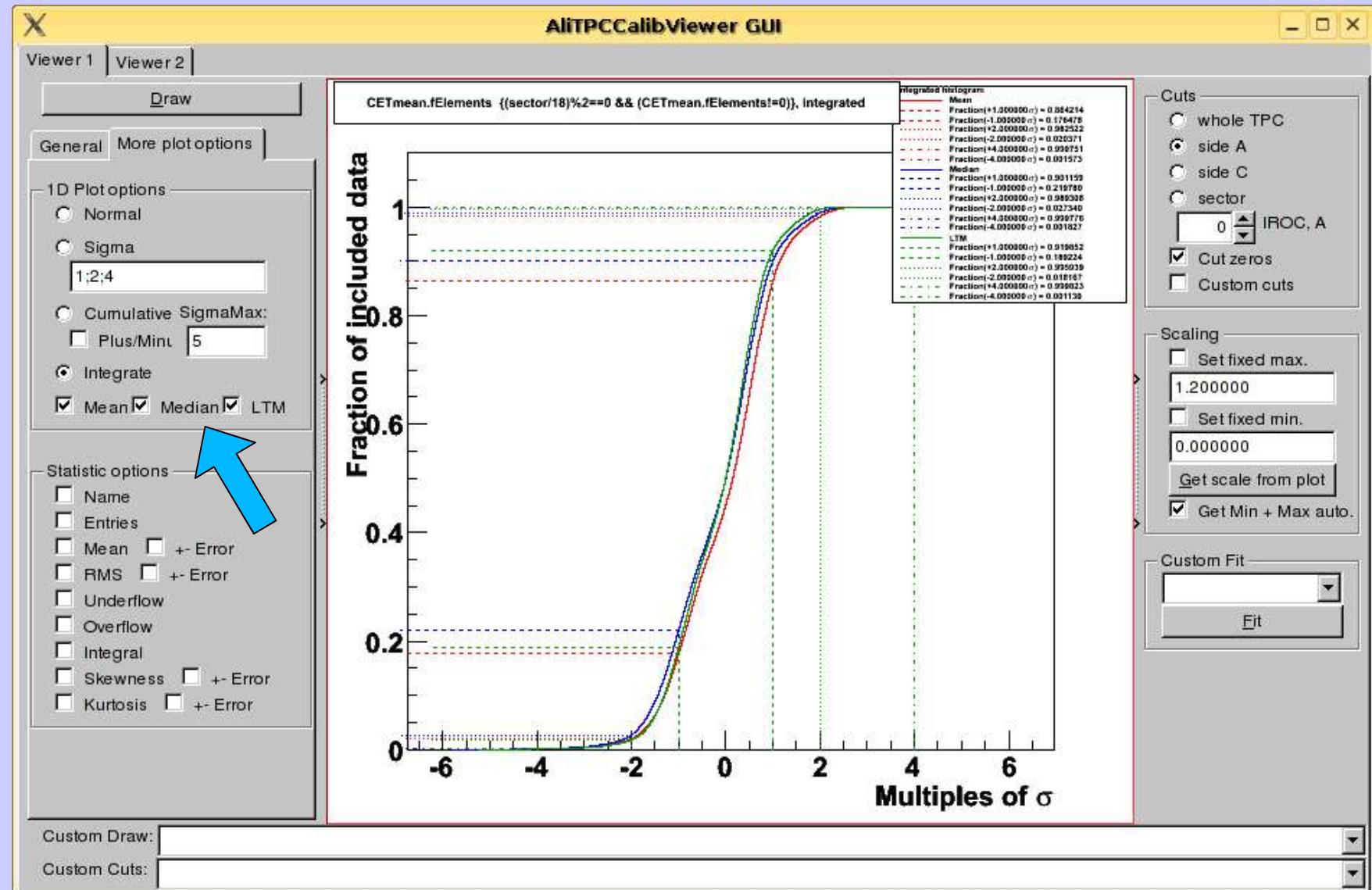
# (Robust) Statistics



# Statistic - Cumulative function



# Statistic – cumulative function

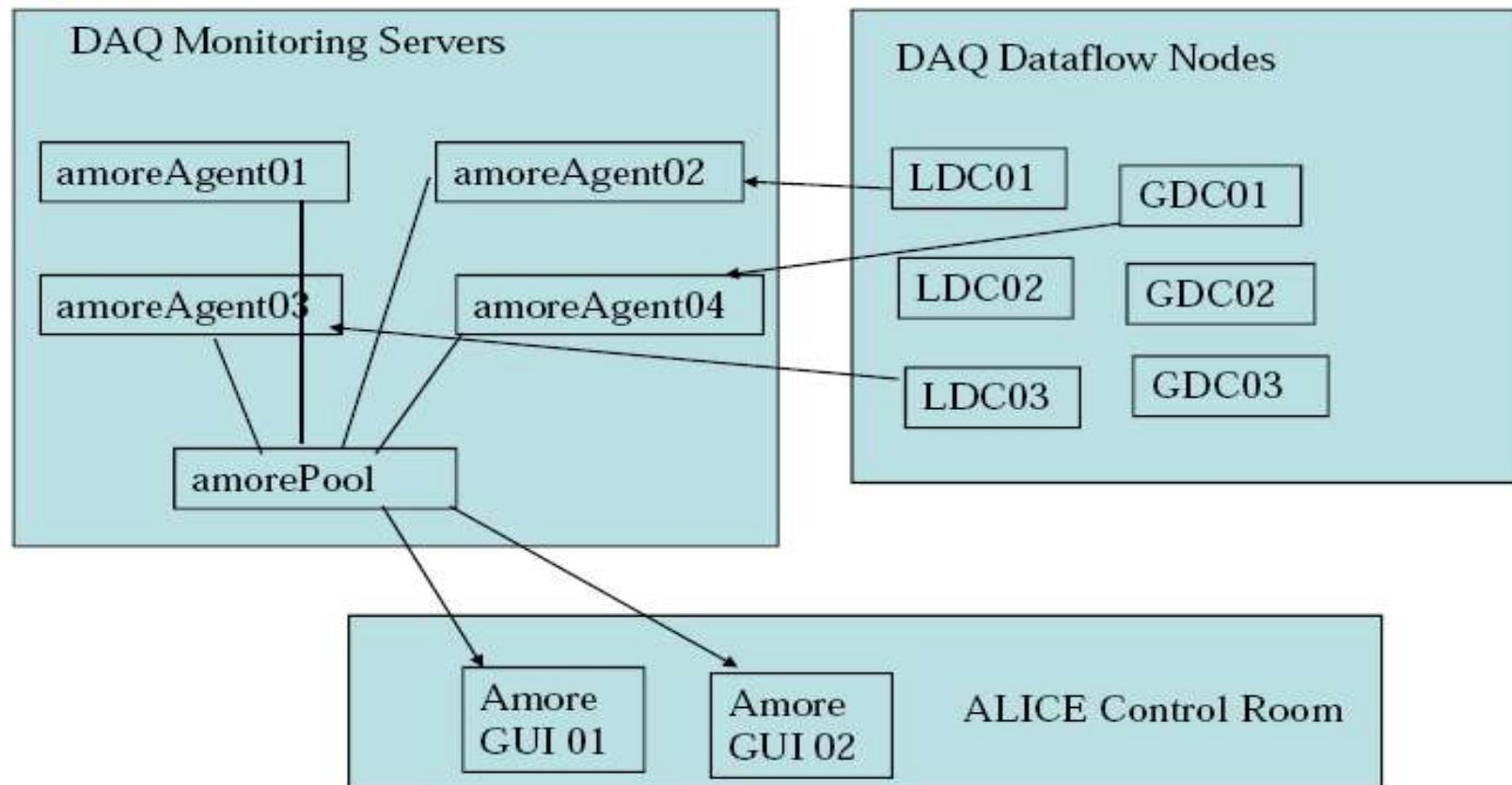


# TPC QA – Amore integration

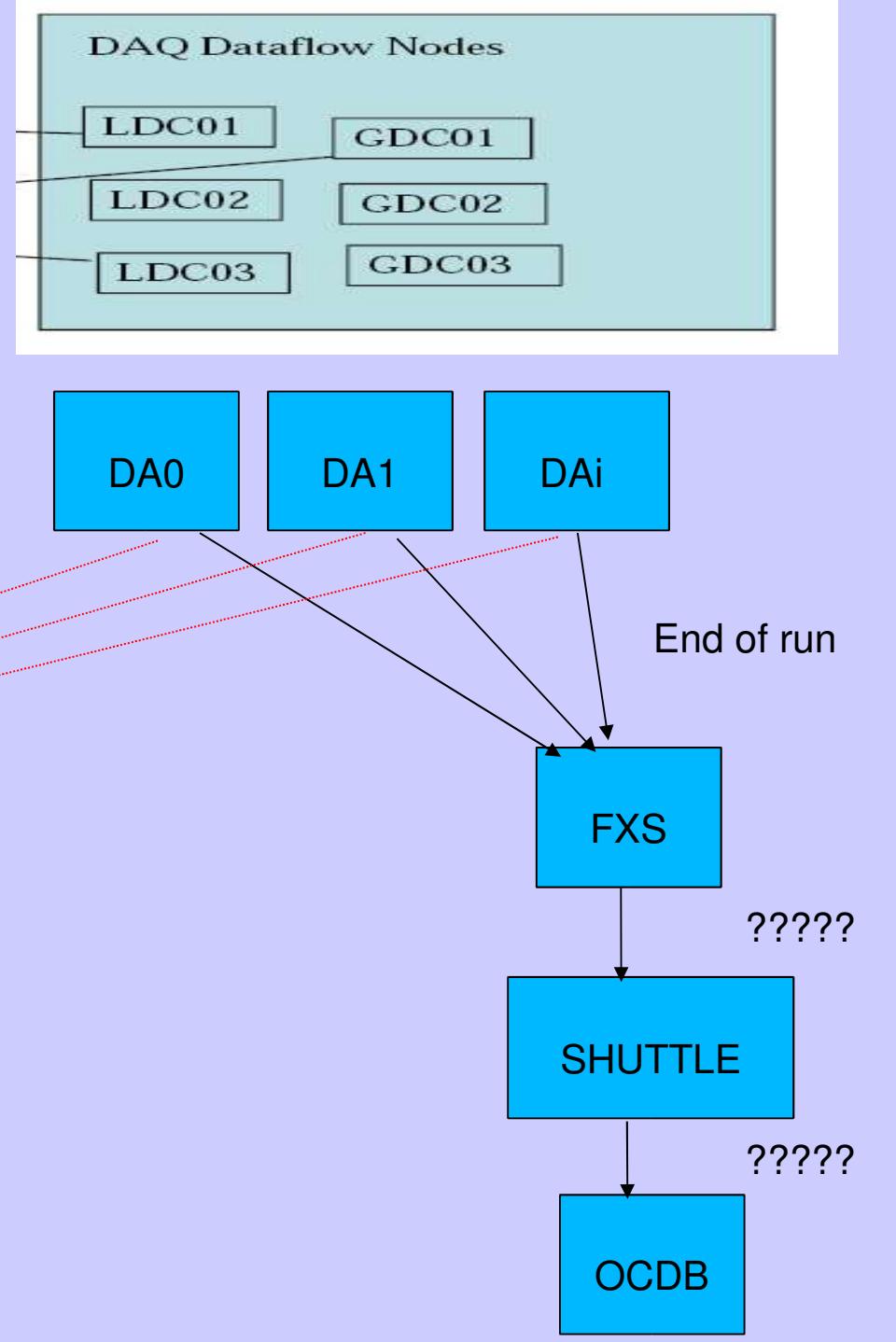
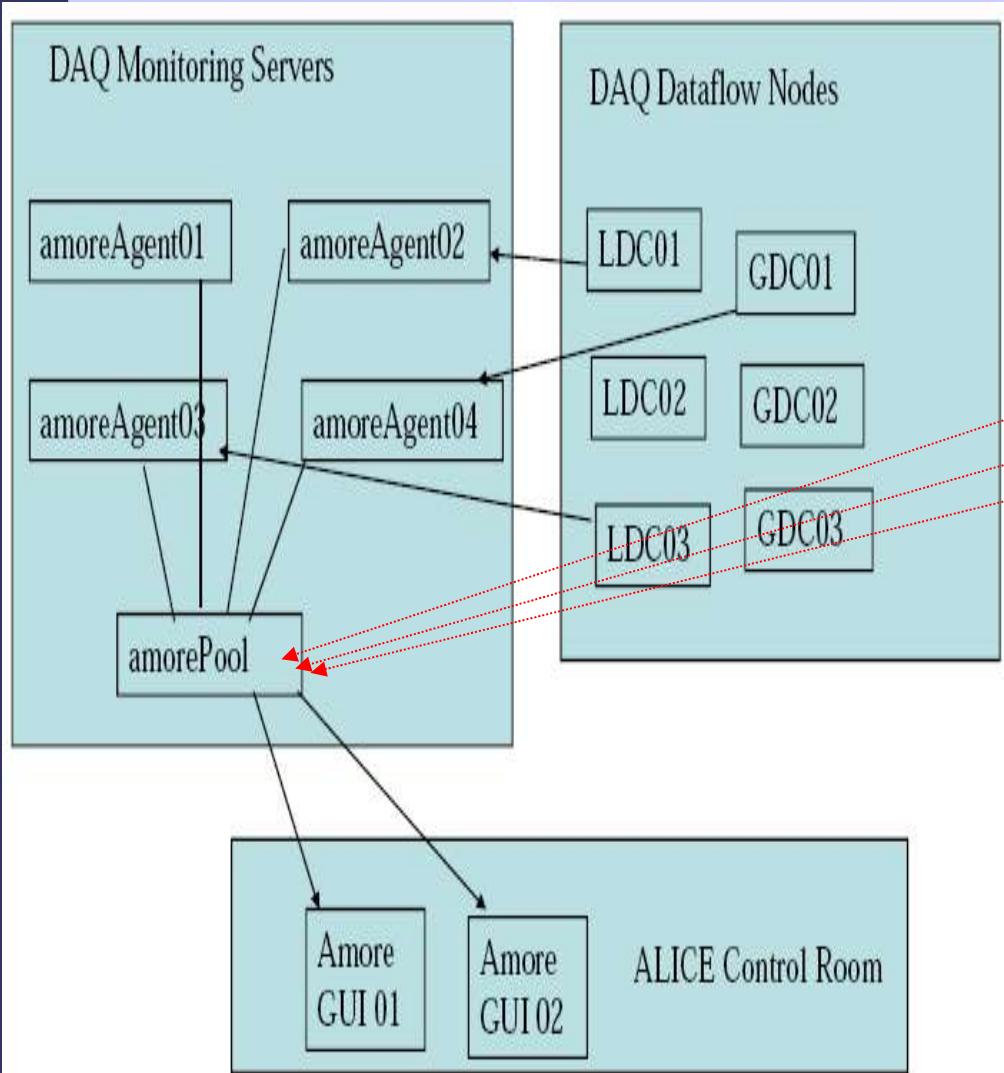
# *Amore - standard*



## The big picture



# *Amore* + DA



# ***TPC implementation***

- All amore related code in SVN
  - AliRoot/TPC/amoreTPC-QA
- Amore QA part – libAmoreTPCPublisher
  - AliTPCDataMakerRec – AliTPCdataQA part
- GUI – libAmoreTPCUI
  - Simple histograms + Expert mode (class AliTPCCalibViewerGUI used also in AliEve)
- Not special development for Amore
  - Amore specific - only data flow

# *DA export implementation (February status)*

- 1. DAs itself - (TPCCEda.cxx.diff)
  - +#include <AmoreDA.h>
  - +int amore::da::Updated(char const\*){return 0;}
  - -+ amore::da::AmoreDA amoreDA;
  - + amoreDA.Send("CE",&calibCE);
- 2. MakefileDA:
  - -CXXFLAGS+=-I\${ALICE\_ROOT}/\$(DAMODULE)
  - +CXXFLAGS+=-I\${ALICE\_ROOT}/\$(DAMODULE) \$(shell amore-config --includes)
  - -DAQDALIB=\$(DAQDADIR)/libdaqDA.a
  - +DAQDALIB=\$(DAQDADIR)/libdaqDA.a \$(AMORE)/lib/libAmoreDA.a \$(ROOTLIBDIR)/mysql.a \${DIMBIN}/libdim.a
  - -CXXFLAGS+=-I\${ALICE\_ROOT}/\$(EXTRADAMODULE)
  - +CXXFLAGS+=-I\${ALICE\_ROOT}/\$(EXTRADAMODULE) \$(shell amore-config --includes)
- Subscriber - GUI part
  - UIQA::GetNoise(){
  - amore::da::AmoreDA amoreDA;

# *TPC Amore integration - Experience and suggestion (0)*

- ◆ Integration of the TPC QA to the AMORE framework – smooth ~ 2 days jobs
  - ◆ Problems only trying to configure the DAQ + AMORE framework on local computer. (Not succeeded)
- ◆ Suggestion
  - ◆ Configure computer with “toy” Amore accessible within CERN network
  - ◆ Otherwise developers forced to develop only in counting room

# *TPC Amore integration - Experience and suggestion (1)*

- ◆ Question
  - ◆ How to use reference data ?
- ◆ Suggestion
  - ◆ Persistent data per run

# *TPC Amore integration*

- ◆ DA – Calibration algorithm – work in progress
  - ◆ Currently just work around
- ◆ More detailed diagnostic – using result of calibration using tracks
  - ◆ Can we include to the on line reconstruction chain ?
  - ◆ Online (mis-)alignment visualization