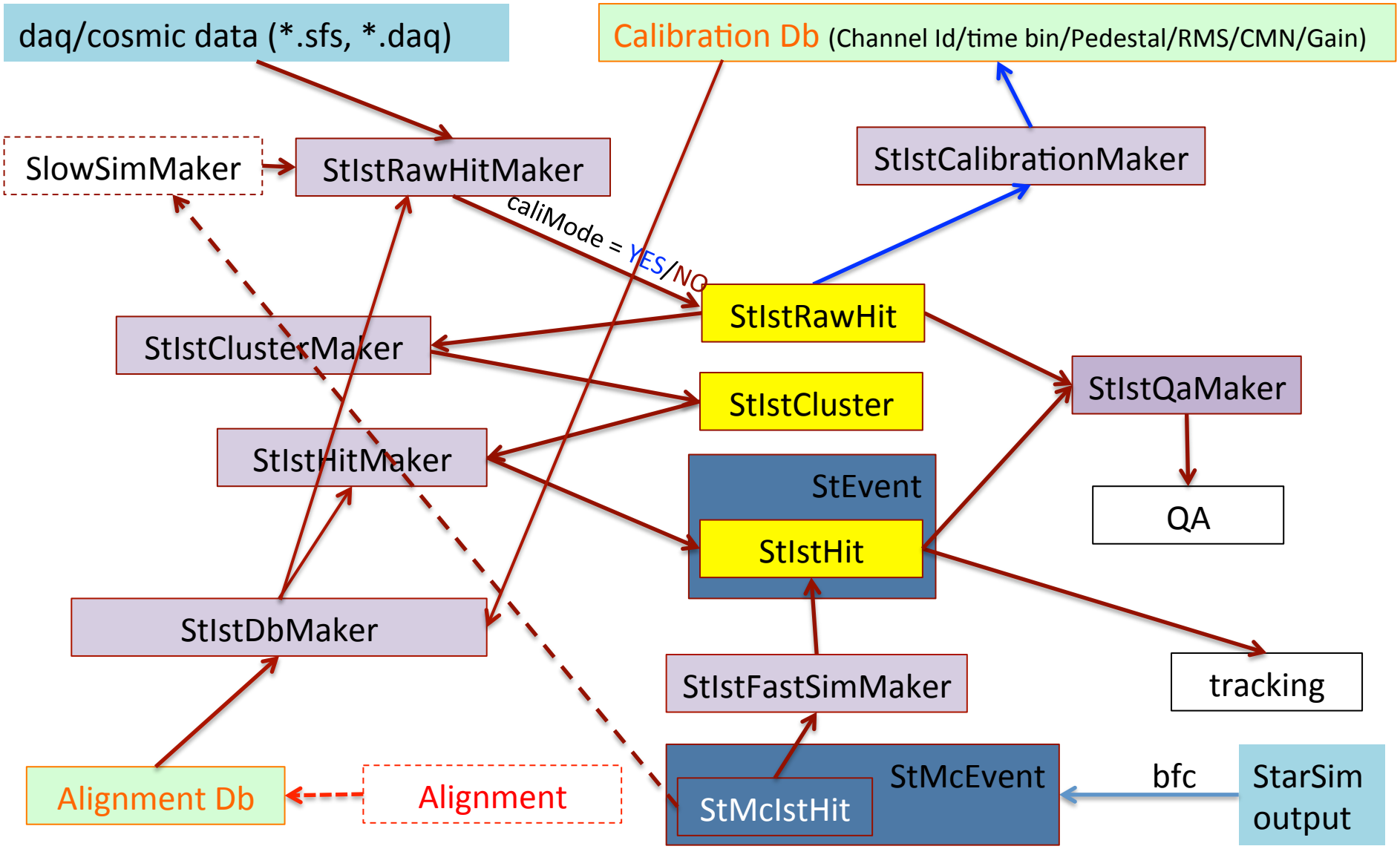
IST Software Status

Yaping Wang
(University of Illinois at Chicago)

Outline

- **Offline software infrastructure**
- **Offline raw hit/cluster/hit structure**
- **Offline makers and IST geometry**
- **Online software status**
- **Summary and Plan**

IST offline chain infrastructure



IST offline chain file list

StRoot/StEvent
- StIstConsts.h
- StIstRawHit
- StIstRawHitCollection
- StIstCluster
- StIstClusterCollection
- StIstCollection
- StIstHit
- StIstHitCollection
- StIstLadderHitCollection
- StIstSensorHitCollection
- StEvent
- StContainers
- StEventClusteringHints

StRoot/StEventUtilities
- StEventHitIter

StRoot/StiRnD/Ist
- StiIstDetectorBuilder
- StiIstHitLoader

StRoot/StIstRawHitMaker
- StIstRawHitMaker

StRoot/StIstClusterMaker
- StIstClusterMaker
- StIstIClusterAlgo.h
- StIstSimpleClusterAlgo

StRoot/StIstHitMaker
- StIstHitMaker

StRoot/StIstDbMaker
- StIstDbMaker

StRoot/StIstCalibrationMaker
- StIstCalibrationMaker

StRoot/StIstQaMaker
- StIstQaMaker (raw hit & hit level)

StRoot/StIstFastSimMaker
- StIstFastSimMaker

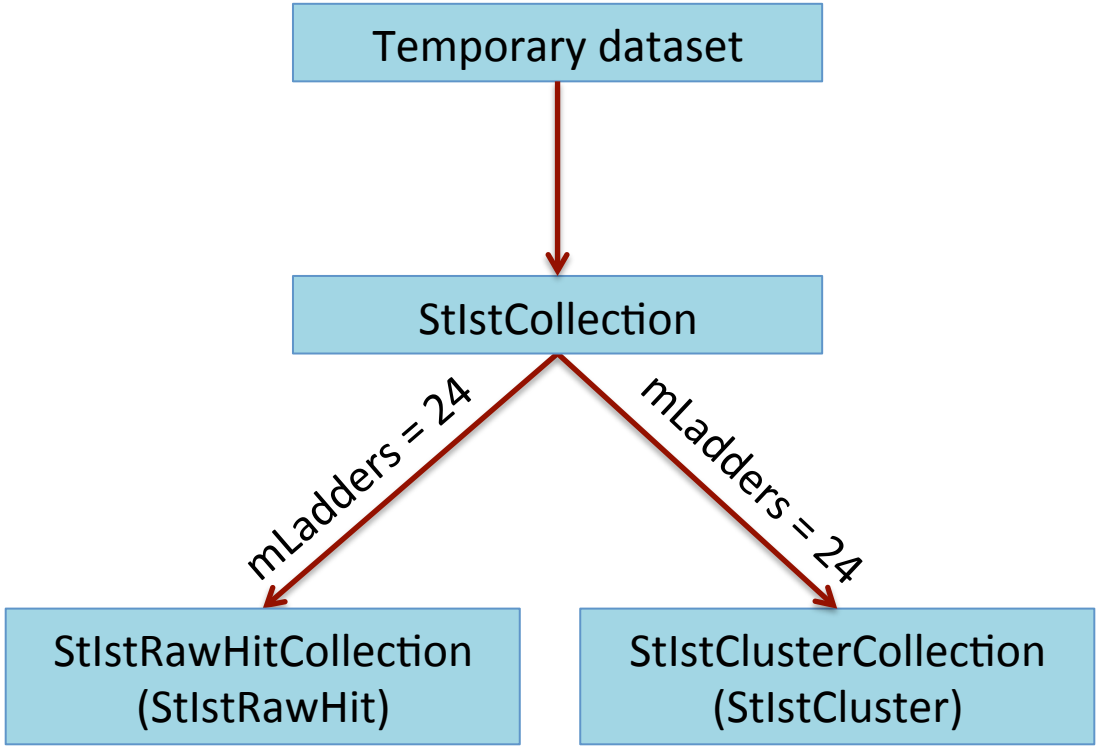
StRoot/StBFChain
- BigFullChain.h

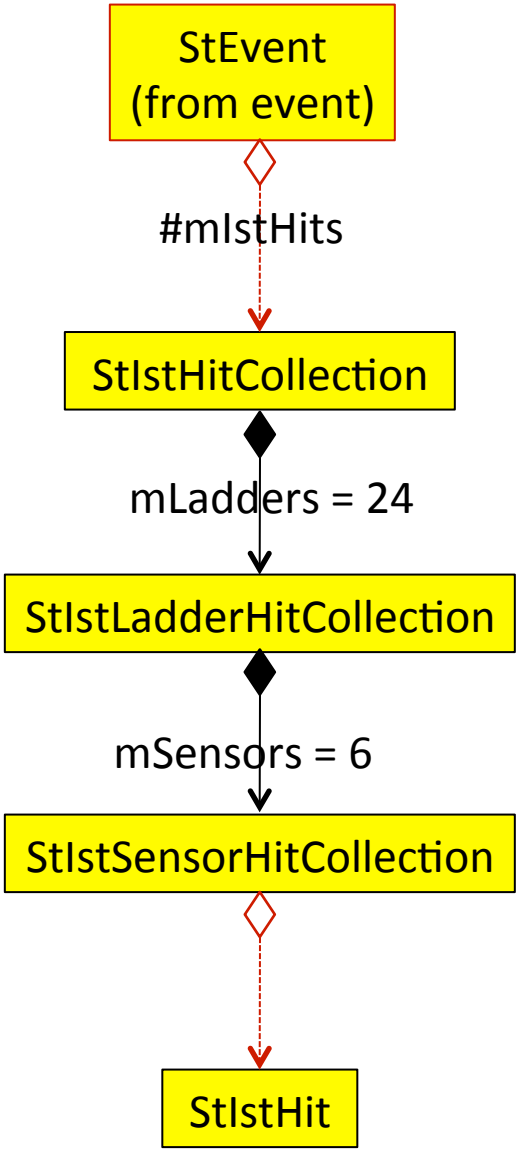
StarDb/Geometry/ist
- idsOnTpc
- pstOnIds
- istOnPst
- istLadderOnIst
- istSensorOnLadder

StarDb/Calibrations/ist
- istPedNoise
- istGain

Wine red: work for IST only
Blue: update for IST

IST offline storage scheme





IST raw hit and cluster information will not be stored into StEvent, and only hit information will be written to the StEvent container.

IST raw hit structure - StIstRawHit

```
    // accessors
    Int_t       getChannelId()  const; //!< 0-110591
    UChar_t     getLadder()     const; //!< 1-24
    UChar_t     getSensor()     const; //!< 1-6
    UChar_t     getRow()        const; //!< 1-64
    UChar_t     getColumn()     const; //!< 1-12
    Float_t     getCharge(UChar_t tb = 0)       const;
    Float_t     getChargeErr(UChar_t tb = 0)    const;
    UChar_t     getMaxTimeBin()                 const;
    UChar_t     getRdo()        const; //!< 1-6
    UChar_t     getArm()        const; //!< 0-5
    UChar_t     getApv()        const; //!< 0-23
    UChar_t     getChannel()    const; //!< 0-127
    UChar_t     getDefaultTimeBin()             const;
    UShort_t    getIdTruth()    const; //!< for embedding, 0 as background

    // modifiers
    void        setChannelId(Int_t rChannelId) ;
    void        setCharge(Float_t charge, UChar_t tb = -1) ;
    void        setChargeErr(Float_t chargeErr, UChar_t tb = -1) ;
    void        setMaxTimeBin(UChar_t tb) ;
    void        setDefaultTimeBin( UChar_t tb )  ;
    void        setIdTruth(UShort_t idTruth);

private:
    // data member
    Int_t       mChannelId;                 // channel Id, numbering from 0 to 110591
    Float_t     mCharge[kIstNumTimeBins];   // raw ADC value saved in calibration mode;
                                            // charge value saved in non-calibration mode (pedestal subtracted and CMN corrected)
    Float_t     mChargeErr[kIstNumTimeBins];// charge error in all time bins
    UChar_t     mMaxTimeBin;                // the max ADC time bin index of the raw hit
    UShort_t    mIdTruth;                   // !< for embedding, 0 as background
    static UChar_t mDefaultTimeBin;
```

IST raw hits in the StIstRawHitCollection container are sorted by geometry ID in increasing order.

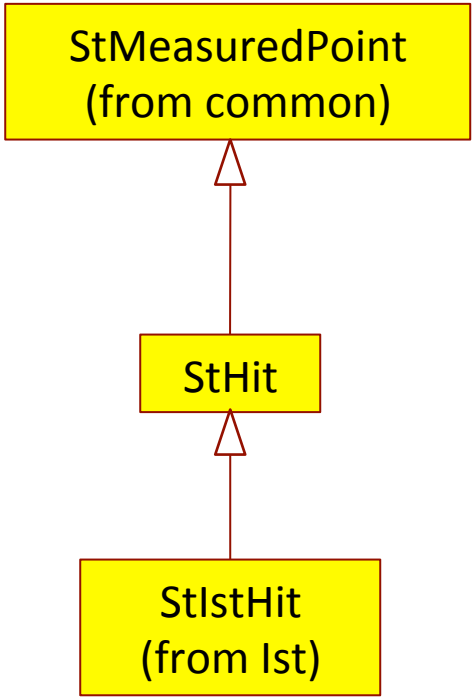
The transform from channel electronics ID (mChannelId) to geometry ID (ladder, sensor, column and row) is done here according to mapping. Ideal mapping is implemented currently, and the real mapping will be updated in 3rd week of October.

IST cluster structure - StIstCluster

```
    //accessories
    Int_t       getKey()                const;
    UChar_t     getLadder()             const;//!< 1-24
    UChar_t     getSensor()             const;//!< 1-6
    Float_t     getMeanRow()            const;
    Float_t     getMeanColumn()         const;
    Float_t     getTotCharge()          const;
    Float_t     getTotChargeErr()       const;
    UChar_t     getMaxTimeBin()         const;
    UChar_t     getClusteringType()     const;
    UChar_t     getNRawHits()           const;
    UChar_t     getNRawHitsRPhi()       const;
    UChar_t     getNRawHitsZ()          const;
    UShort_t    getIdTruth()            const;

    //modifiers
    void        setLadder(UChar_t ladder);
    void        setSensor(UChar_t sensor);
    void        setMeanRow(Float_t meanRow);
    void        setMeanColumn(Float_t meanColumn);
    void        setTotCharge(Float_t totCharge);
    void        setTotChargeErr(Float_t totChargeErr);
    void        setMaxTimeBin(UChar_t tb);
    void        setClusteringType(UChar_t clusteringType);
    void        setNRawHits(UChar_t nRawHits);
    void        setNRawHitsRPhi(UChar_t nRawHitsRPhi);
    void        setNRawHitsZ(UChar_t nRawHitsZ);
    void        setIdTruth(UShort_t idTruth);

protected:
    Int_t       mKey;                       //cluster unique lable
    UChar_t     mLadderId;                  //ladder id the cluster belongs to
    UChar_t     mSensorId;                  //sensor id the cluster belongs to
    Float_t     mMeanRow, mMeanColumn;      //mean row and mean column
    Float_t     mTotCharge;                 //charge sum of the cluster
    Float_t     mTotChargeErr;              //RMS noise of the cluster
    UChar_t     mMaxTimeBin;                //max ADC time bin index
    UChar_t     mClusteringType;            //clustering algorithm type
    UChar_t     mNRawHits;                  //cluster size
    UChar_t     mNRawHitsRPhi;              //cluster size in r-phi direction
    UChar_t     mNRawHitsZ;                 //cluster size in beam direction
    UShort_t    mIdTruth;                   //!< for embedding, 0 as background
    rawHitMap_t mRawHitMap;                 //map container to save raw hits who contribute to the cluster
```

IST hit structure - StIstHit

```
    //accessories
    UChar_t     getLadder() const;
    UChar_t     getSensor() const;
    Float_t     getMeanColumn() const;
    Float_t     getMeanRow() const;
    UChar_t     getMaxTimeBin() const;
    Float_t     getChargeErr() const;
    UChar_t     getClusteringType() const;
    UChar_t     getNRawHits() const;
    UChar_t     getNRawHitsZ() const;
    UChar_t     getNRawHitsRPhi() const;
    Float_t     localPosition(UInt_t ) const;
    Float_t     localPositionErr(UInt_t ) const;

    //modifiers
    void    setMaxTimeBin(UChar_t tb);
    void    setChargeErr(Float_t chargeErr);
    void    setClusteringType(UChar_t clusteringType);
    void    setNRawHits(UChar_t nRawHits);//cluster size
    void    setNRawHitsZ(UChar_t nRawHitsZ);
    void    setNRawHitsRPhi(UChar_t nRawHitsRPhi);
    void    setLocalPosition(Float_t, Float_t, Float_t);

protected:
    UChar_t mMaxTimeBin;        // max charge time bin
    Float_t mChargeErr;         // charge uncertainty
    UChar_t mClusteringType;    // clustering algorithm type
    UChar_t mNRawHits;          // nRawHits: cluster size
    UChar_t mNRawHitsZ;         // cluster size in Z direction
    UChar_t mNRawHitsRPhi;      // cluster size in r-phi direction

    //local position of hit inside the sensor
    Float_t mLocalPosition[3]; //meanColumn, 0. or defined by sensor surface correction, meanRow

    StDetectorId mDetectorId;
```



IST raw hit maker - StIstRawHitMaker

```
Int_t rdo=0; // rdo numbering from 1, 2, ..., 6
Int_t arm=0; // arm numbering from 0, 1, ..., 5
Int_t apv=0; // apv numbering from 0, 1, ..., 23
Int_t channel=0; // channel numbering from 0, 1, ..., 127
Int_t adc=0;
Short_t timebin=0;
```

```
mRdr->get(0,EVP_TYPE_ANY);
if(mRdr->status == EVP_STAT_EOR) {
    LOG_DEBUG <<"End of File reached..."<<endm;
    return kStEOF;
}
```

```
daq_dta *dd = 0;
if(mDataType==0){
    dd = mRdr->det("fgt")->get("adc");
    if(!dd) dd = mRdr->det("fgt")->get("zs");
} else if(mDataType==1){
    dd = mRdr->det("fgt")->get("adc");
} else if(mDataType==2){
    dd = mRdr->det("fgt")->get("zs");
} else if(mDataType==3){
    dd = mRdr->det("fgt")->get("zs");
    if(!dd) dd = mRdr->det("fgt")->get("adc");
}
```

This part is copied from FGT daq reader and is used for IST prototype cosmic data analysis. Waiting for IST DAQ reader…

1) Daq data Reader (currently use fgt daq_dta)

2) Accessing calibration DBs via StIstDbMaker and get the pedestal/rms noise/gain for each channel and common mode noise for each chip

3) Mark out bad chips/channels

4) Create IST raw hit, and two working modes included:

- Calibration mode: only write ADCs and channel ID to StIstRawHit
- Physics mode: pedestal subtraction, CM noise correction on/off, signal-like raw hit decision, and write the signal-like raw hit full information to StIstRawHit

IST pedestal/noise maker - StIstCalibrationMaker

STAR HFT

Pedestal/RMS noise data file

Channel Id	time bin	pedestal	rms noise
0	0	760.432	20.4917
0	1	770.887	19.881
0	2	766.958	19.2735
0	3	769.477	19.9935
0	4	768.036	20.2071
0	5	770.344	21.0541
0	6	766.367	22.7797
1	0	800.794	17.4194
1	1	810.621	16.6912
1	2	806.322	16.8651

Common mode noise data file

Chip Id	time bin	cm noise
0	0	9.20214
0	1	7.14334
0	2	10.9961
0	3	8.5315
0	4	10.2901
0	5	10.5285
0	6	13.5949
1	0	8.0494
1	1	6.52007
1	2	6.98871

- Calculate pedestal/rms noise by histogram method for each channel over all time bins
  - Broken/noisy channels can be marked out by setting high rms noise values (1000)
- Calculate common mode noise for each APV chip over all time bins
  - Bad/Strange chips can be marked out by setting high CM noise value (1000)
- Two calibration data files are generated, which are used to write calibration DBs.
- Several QA histograms (pedestal/ rms noise/gain vs. channel Id + cm noise vs. chip Id) are generated

Currently, the gain calculation maker is not available yet, and will be available later. The gain for each channel is set to 1.0 in the current calibration maker.

IST cluster maker - StIstClusterMaker

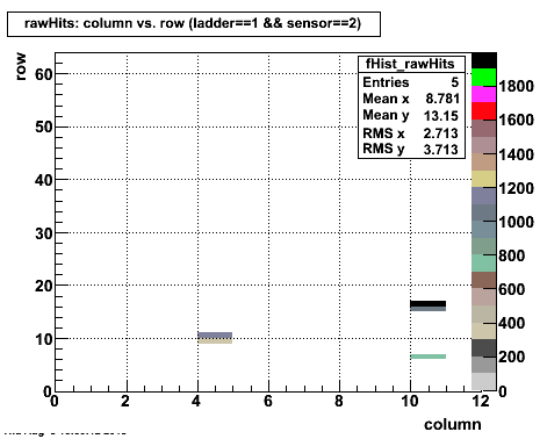
The StIstClusterMaker executes the raw hits clustering in a sensor area ladder by ladder.

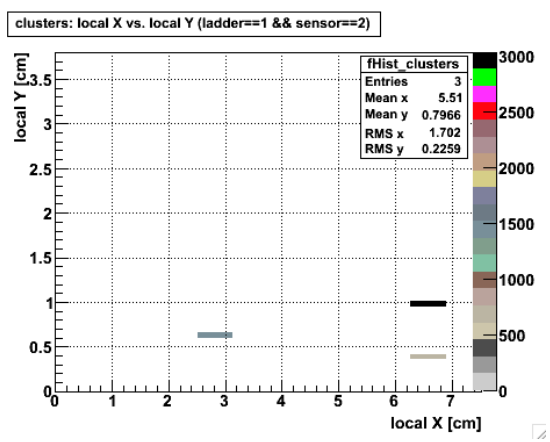
An interface, “Int_t setClusterAlgo(StIstIClusterAlgo*)” is left in the StIstClusterMaker to implement different algorithms, and each algorithm should be inherited from StIstIClusterAlgo virtual class.

$$ADC_{cluster} = \sum_{i=1}^{N_{pads}} ADC_i \quad , \quad \sigma_{cluster} = \sqrt{\sum_{i=1}^{N_{pads}} \sigma_i^2 / N_{pads}}$$

$$x_{cluster} = \sum_{i=1}^{N_{pads}} x_i \cdot w_i \quad , \quad y_{cluster} = \sum_{i=1}^{N_{pads}} y_i \cdot w_i$$

$$w_i = ADC_i / \sum_{i=1}^{N_{pads}} ADC_i$$





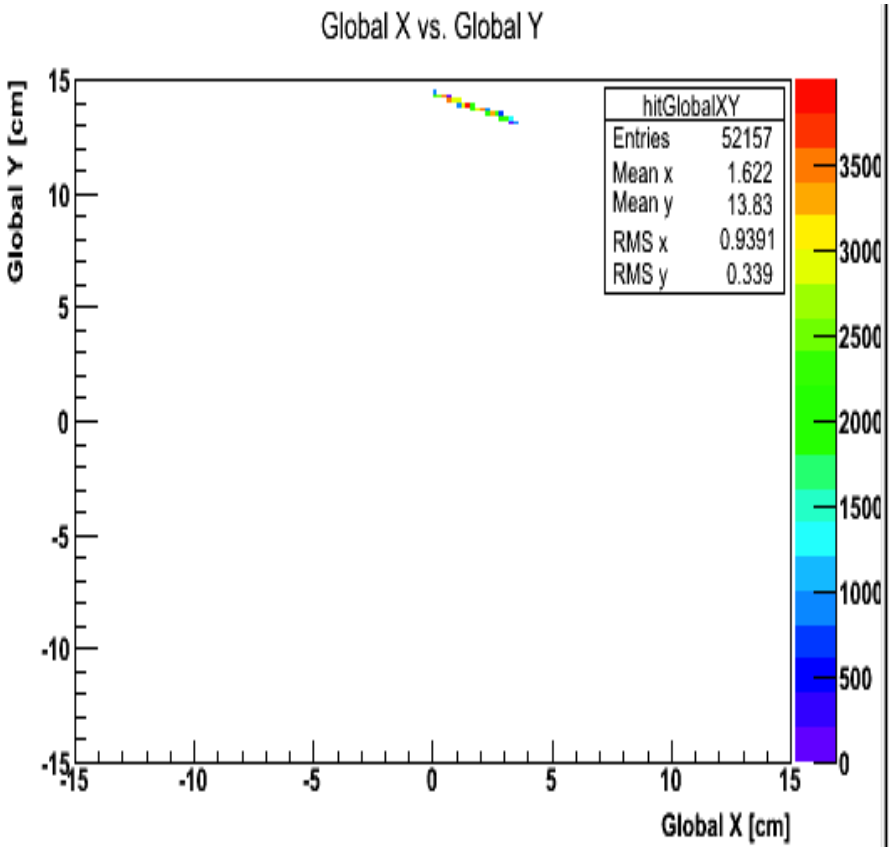
StIstSimpleClusterAlgo class is available for raw hits clustering and splitting:

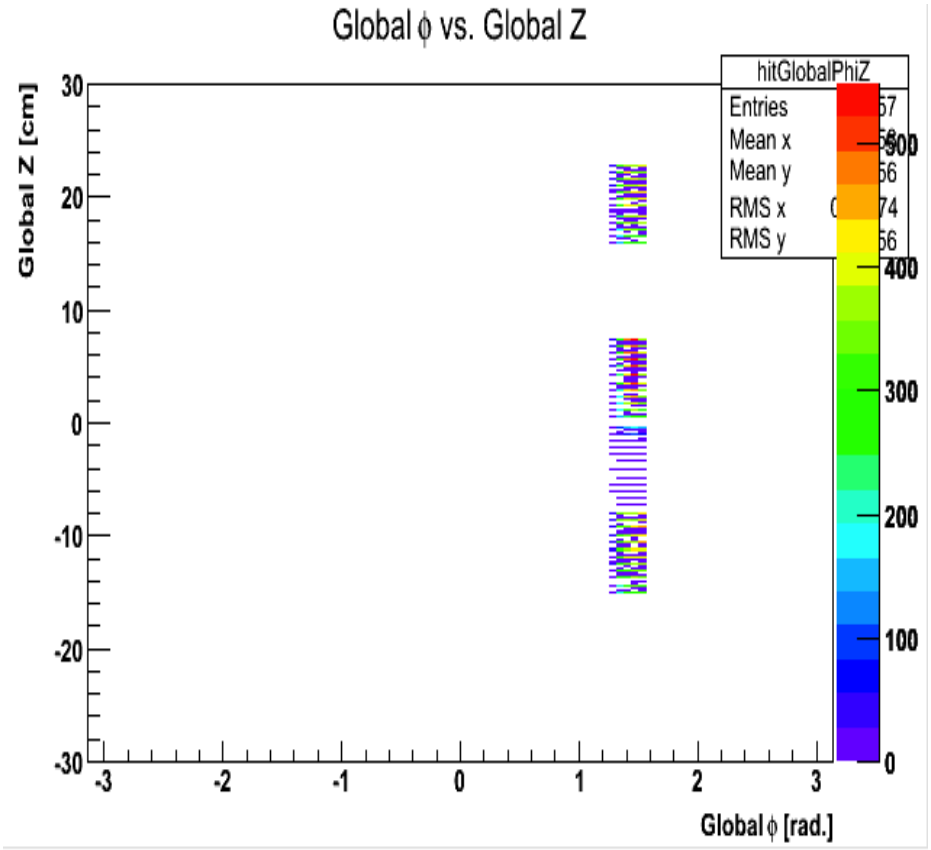
- Read all raw hits per ladder (six sensors) into a vector (sorted by geoID in increasing order)
- Staring from the 1st raw hit, loop the vector to look for neighboring raw hits (in a sensor area) and clustering
- A case-by-case splitting algorithm can be enabled/ disabled for the found clusters splitting (here only works for cases with cluster size ≤ 4)

A new clustering algorithm development is ongoing …

IST hit maker - StIstHitMaker

- Create IST hits according to clusters info.
- Access IST geometry DBs via StIstDbMaker, and get geometry matrix list by:
  gStIstDbMaker->GetRotations()
- Transform local to global position by:
  geoMSensorOnGlobal->LocalToMaster(local, global)





IST DB maker - StIstDbMaker

- Read calibration DB tables in StIstRawHitMaker to get pedestal/rms noise/gain per channel and common mode noise per chip, also mark out bad behavior chips/channels

- Read geometry DB tables in StIstHitMaker to calculate hit global position.

```
//_____________________________________________________________________
Int_t StIstDbMaker::InitRun(Int_t runNumber) {
  CalculateSensorsPosition();
  LOG_DEBUG <<" StIstDbMaker::InitRun() --> GetIstPedNoise" << endm;
  GetIstPedNoise();
  LOG_DEBUG <<" StIstDbMaker::InitRun() --> GetIstGain" << endm;
  GetIstGain();
  return kStOK;
}
```

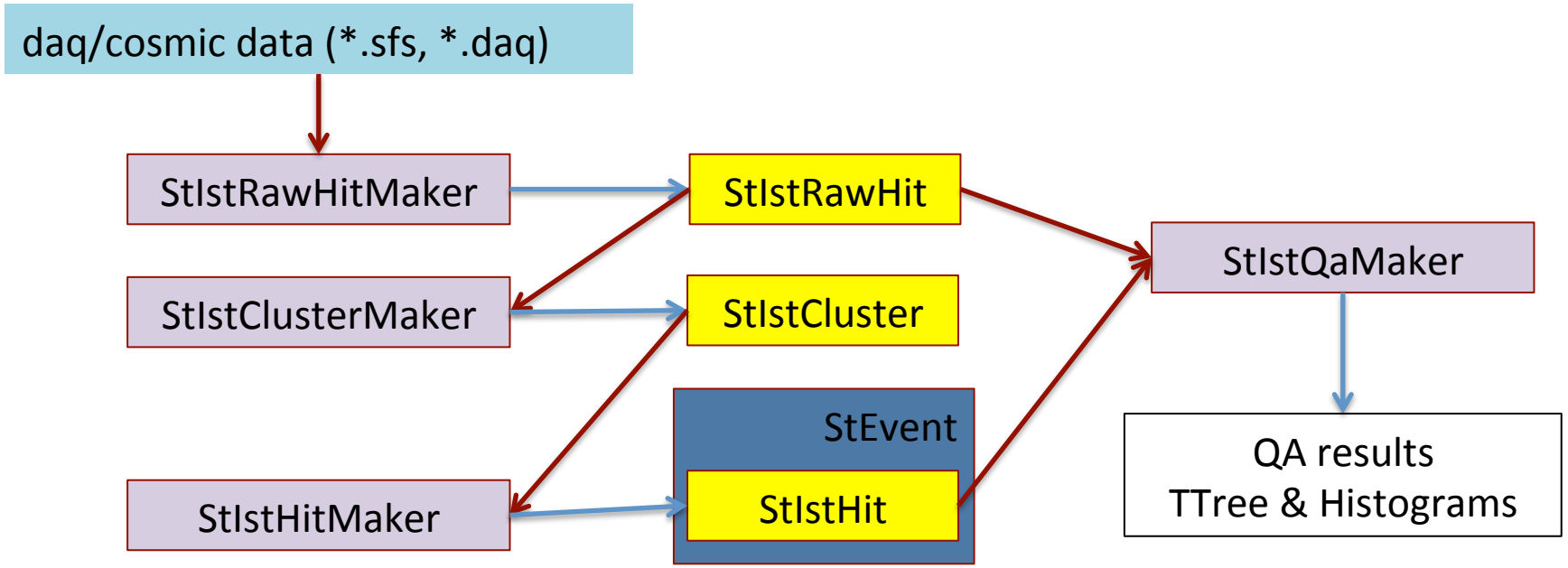
IST geometry/calibration DBs:

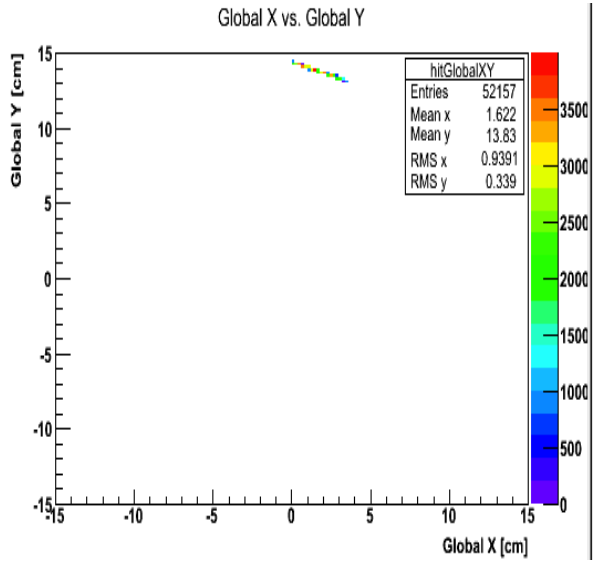
- Two databases created: http://online.star.bnl.gov/dbExplorer/
  - Geometry_ist: istSurvey, istOnPst, istLadderOnIst, istSensorOnLadder, idsOnTpc, pstOnIds
  - Calibration_ist: istGain, istPedNoise

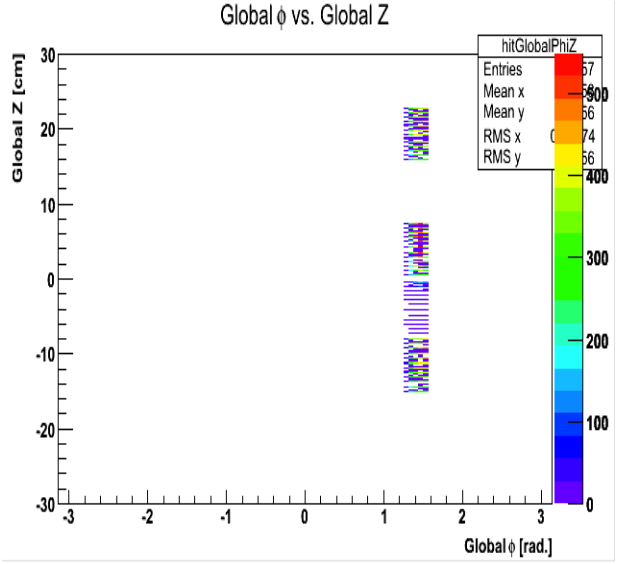
- Getting the IST survey data into geometry DBs will be started in 4th week of October

- Calibration DBs population will be started in the November cosmic run.

IST offline QA maker - StIstQaMaker

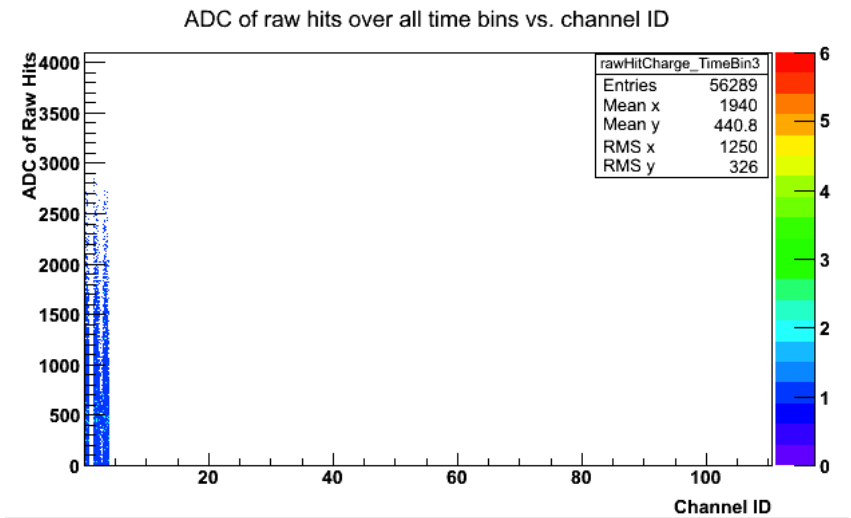




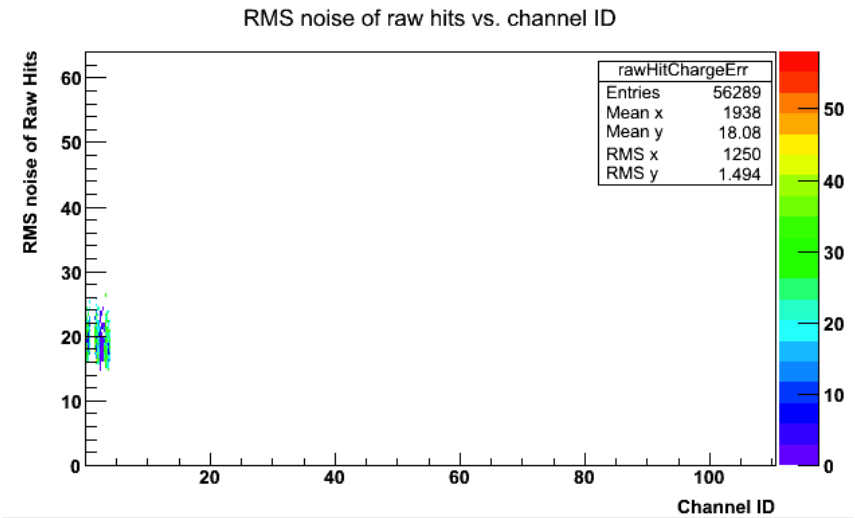


IST offline QA maker - StIstQaMaker

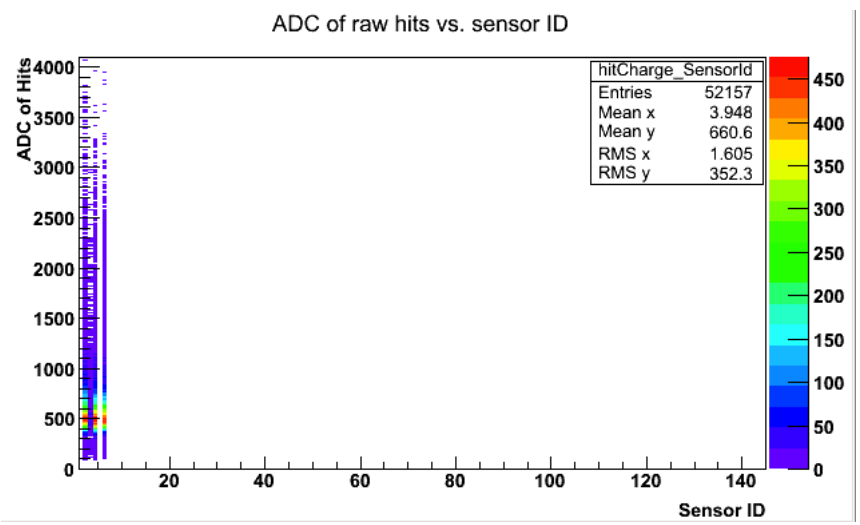
Raw hit ADC vs. channel ID (time bin = 3)



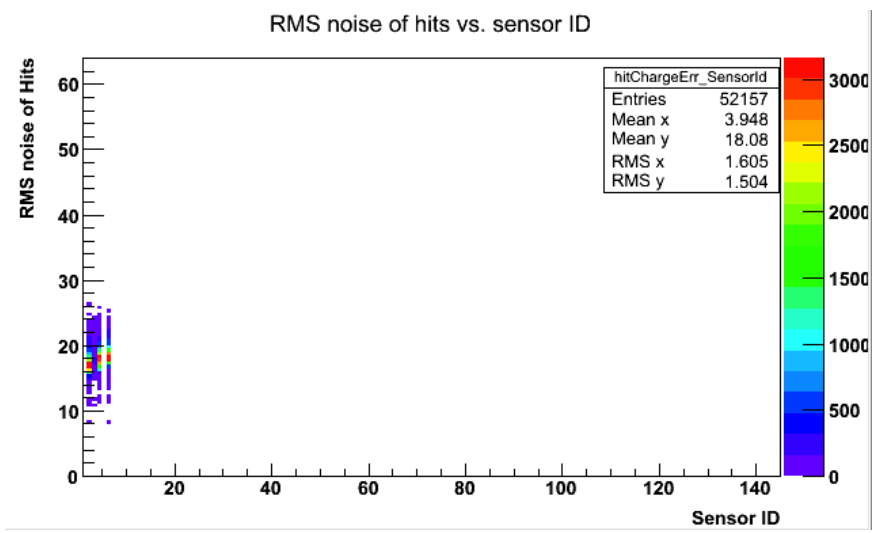
Raw hit RMS noise vs. channel ID (all channels)



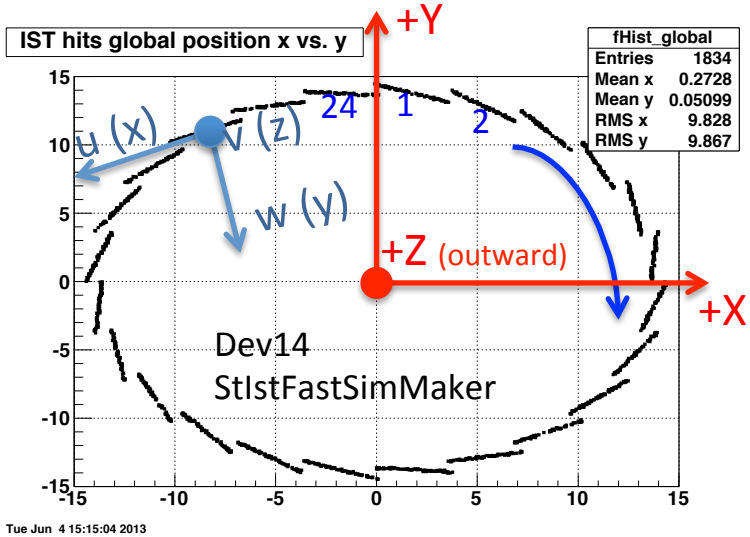
Hit ADC vs. sensor ID



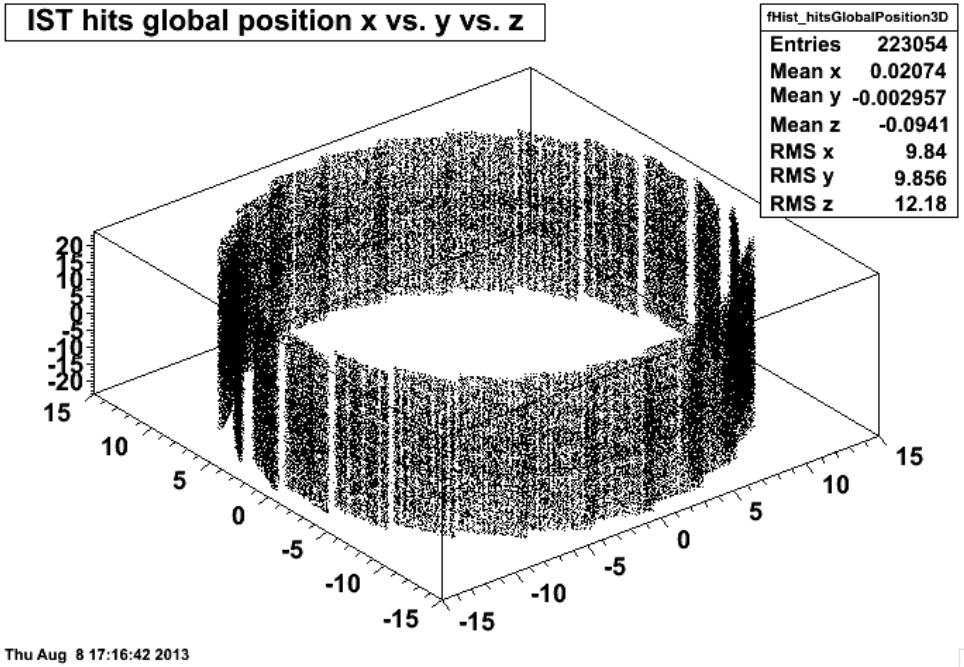
Hit RMS noise vs. sensor ID



IST fast simulation maker – StIstFastSimMaker



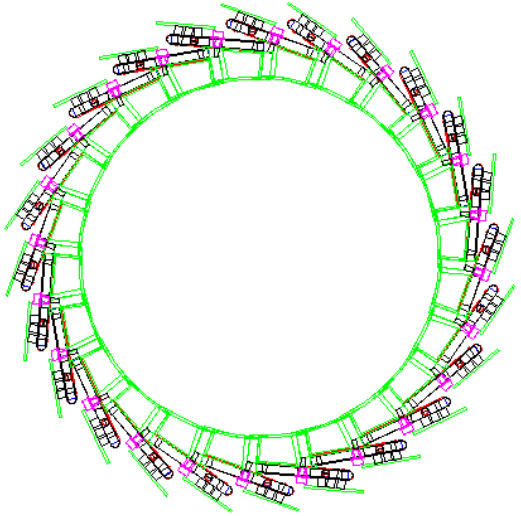
1. New StIstHit data structure and IST offline chain are applied.
2. StIstDetectorBuilder and StIstHitLoader are updated to use the new offline/hit structure
3. StBFChain/BigFullChain.h updated (add "istFastSim" option)



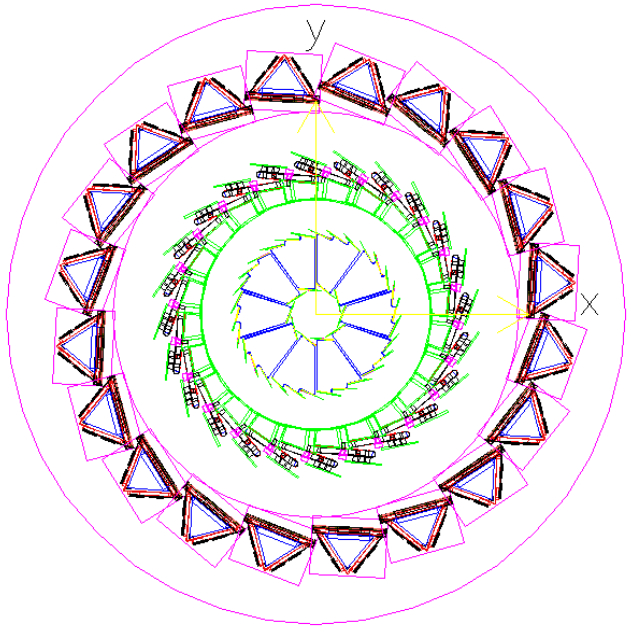

```
detp geom dev14
gexec .$STAR_HOST_SYS/lib/xgeometry.so
gclose all

gfile o 48pion_5000evts.fz
user/vsig 0 .1
aguser/gkine 48 8 0.5 5.0 -1.5 1.5 0 6.28
trig 5000
exit
```

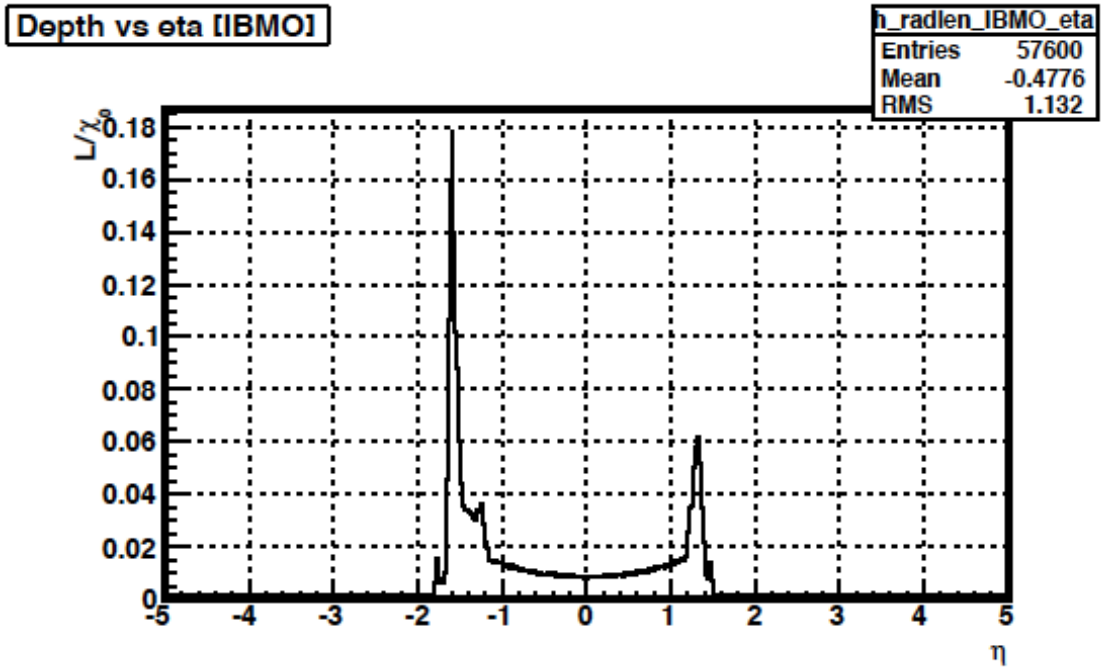
root4star -b -q 'StRoot/macros/bfc.C(1,5000,"dev14,AgML,ITTF,Sti,tpcI,TpcFastSim,istFastSim,-pxlFastSim,-ssdfast, VFMCE,McEvent,geant, IdTruth,fzin,StiRnD,-PixelIT,IstIT,NoSvtIt,NoSsdIt,-StiPulls,analysis,-tags,clearmem,evout,McEvOut,MiniMcMk,McAna,MakeEvent","/star/u/ypwang/disk02/istOfflineChain/offlineChain20130617/48pion_5000evts.fz")'

IST GEANT geometry – dev14



1. Detailed modeling for IST ladder structure
2. The dev14 is included in full HFT simulation chain with the new IST offline chain.
3. To do:
   1) support blocks update (shape changed)
   2) cabling/cable fixing rings/cooling lines modeling





IST online software

- The slow control software is available, but need to be modified to shift operations (ongoing): **Chanaka** and **Yadav**

- Online monitoring software needs to develop, and will be started after deciding zero suppression to be executed in FPGA level or in DAQ-PC level: **Yaping** and **Zillay**

Summary and Plan

STAR HFT

- The IST offline software is mostly finished and tested with IST prototype cosmic data (.sfs)
- The IST offline software is almost ready for review.
- Implement the new clustering algorithm in 3rd week of October.
- Get survey data to geometry DBs in 4th week of October.
- Populate Calibration DBs during cosmic runs.
- Online software is in progress, and will be available before November cosmic run.
- IST online/offline software testing with full barrel readout will be done in cosmic runs.