HCAL Software Status

CALICE meeting, September 11th 2007

Sebastian Schmidt, DESY

Overview

- General ILC software issues in the view of the HCAL software developers
- Review of the developments since February
- New integrated HCAL reconstruction and digitization
- Outlook

ILC software issues

- We (Roman, Frank, S.) performed tests of IO performance of LCIO and binary format
- Reading a file in the binary format takes about as long as `cat > /dev/null`
- Reading an LCIO file takes considerably longer (factor depends on event size, cpu, disk performance)
- Findings
  - LCIO independent of byte ordering and 32/64 bit architecture
  - Every LCIO collection is zipped
  - Performance decrease due to database accesses negligible
  - LCIO: All data are stored in objects (i.e. copied in the memory at least once after reading in)
    - All parts of an event can be used simultaneously directly after reading in
  - Binary format: For every DAQ record read the same buffer in memory is filled, interpreted and then overwritten again
    - When using the binary format calibrations/ zero suppressions can be performed without event building
- Software developers are aware of performance issues of LCIO
- But: For modular software development (i.e. more than one person working) one needs some kind of interfaces between the parts
- LCIO is the best framework defining those interfaces available
- One can however play some tricks:
  - Avoid collections with lots of entries but use collections of vectors instead (saves time but ignores interfaces)
  - After finalizing all calibration steps $\rightarrow$ put everything in a single processor

Previous releases of HCAL software

- hcal-v00-01-18 available early March
- Muon finder
- Extended set of calibrations
  - Set of conditions data which extrapolates existing “approved” calibrations to all time ranges for which it makes sense wrt HV setting: replace “Approved” by “Inofficial” in the path to the conditions data to make use of it
  - Useful for non HCAL analysis (like TCMT) or even HCAL calibration studies (muon runs!)
  - Excellent example of how to use conditions data! (extract time ranges and HV for all runs from data base and assign calibration data accordingly …)

Previous releases of HCAL software

- hcal-v00-01-19 available since end of May
- Introduced interface for temperature dependent calibrations
  - Temperature dependent gain calibration available from S. Schätzel
  - Not yet implemented: temperature dependent MIP calibrations, but trivial if one has the example of the gain calibration
- CED is installed during the installation of the package as alternative event display (volunteers needed for a data base dependent geometry display)
- Unofficial version of the tail catcher reconstruction (without any data base usage) by Guilherme
- Interface to cluster algorithm (“deep analysis”) by Vassilly (seems to deliver reasonable clusters, details still have to be checked!)
- Update of the LED tools (S. Schätzel)

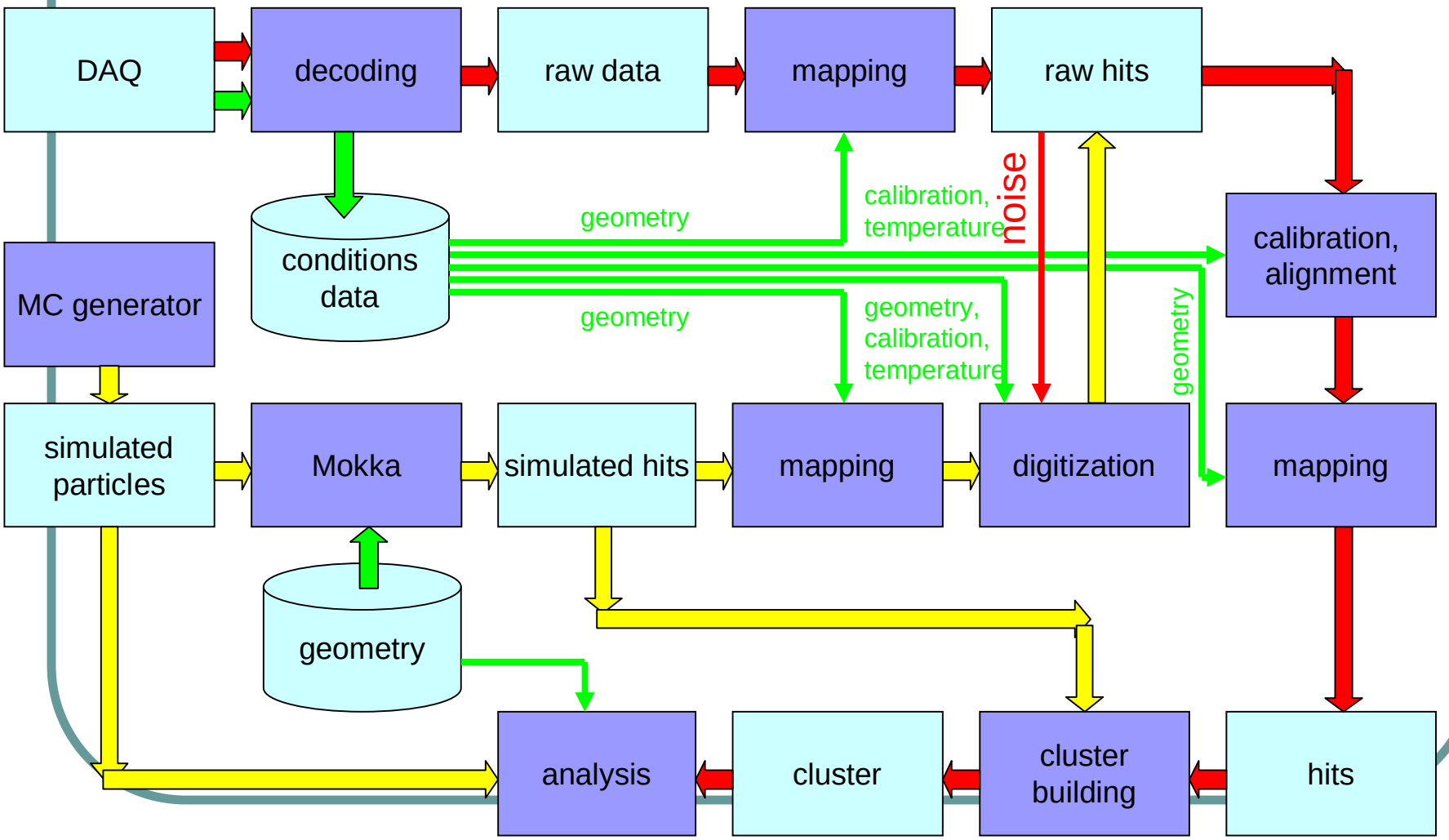
Current release of HCAL software

- hcal-v00-01-20 available since end of June
- Provides complete reconstruction and digitization for the HCAL
  - Relies completely on the data base for calibration and geometry
  - Reimplements proprietary codes by Niels and Nanda
  - Reconstruction has been checked by Nanda, digitization still to be checked (Niels)
  - Some fixes and extensions are already available on request
  - Ready to be implemented in official reconstruction and digitization, follows official data streams (no flat files, no conversion of MC files to flat files, etc.), allows parallel reconstruction/digitization of ECAL/HCAL/… in a single job
  - Based on inherited processors
- Steering files for data, MC and extraction of noise files available from the web

Current release of HCAL software (cont'd)

- Coarse modules for 2007 test beam available, cell sizes are in the data base now
- Steering file for preliminary reconstruction (only MIP calibration) of 2007 data available
- Correction in the HCAL temperature model for stuck temperature sensors in 2006
- Considerable performance improvements
  - It's quite interesting in what places of C++ code the performance problems occur (memory management, exceptions, …)
  - New ECAL reconstruction uses up some of the gain in speed by the HCAL ☹
  - For SL3 users: It might make sense to upgrade to SL4 as the latest LCIO version induces a heavy performance drop of O(25%) together with the STL version installed on SL3

CALICE (HCAL) data streams



Basic design of HCAL reconstruction/digitization

- Reconstruction
  - Mapping from ADC channels to hardware channels
    - Using the geometry in the data base
  - Apply calibrations (one or more processors)
    - Using the calibrations (MIP, gain, intercalibration) in the data base
  - Mapping from hardware channels to physics channels
    - Using the geometry in the data base
- Digitization
  - Distribution of events over specific run period
  - Mapping from physics channels to hardware channels
    - Using the geometry in the data base
  - Apply digitization, add noise
    - Neighbor relations are taken from the geometry in the data base
    - Noise is read in from noise file (lcio) generated from pedestal events
    - Inverse of calibrations is used for digitization
    - Statistical smearing is applied to # of pixels
  - Apply calibrations
    - Exactly like for the reconstruction
  - Mapping from hardware channels to physics channels
    - Using the geometry in the data base
- Implementation ensures that ECAL, HCAL, TCMT, drift chamber reconstruction and digitization can run in the same job
- Internal data type between the processors is FastCaliceHit

FastMappingIProcessor

IntegratedHcalReconstructionProcessor

FastMappingIIProcessor

RunInfoProcessor

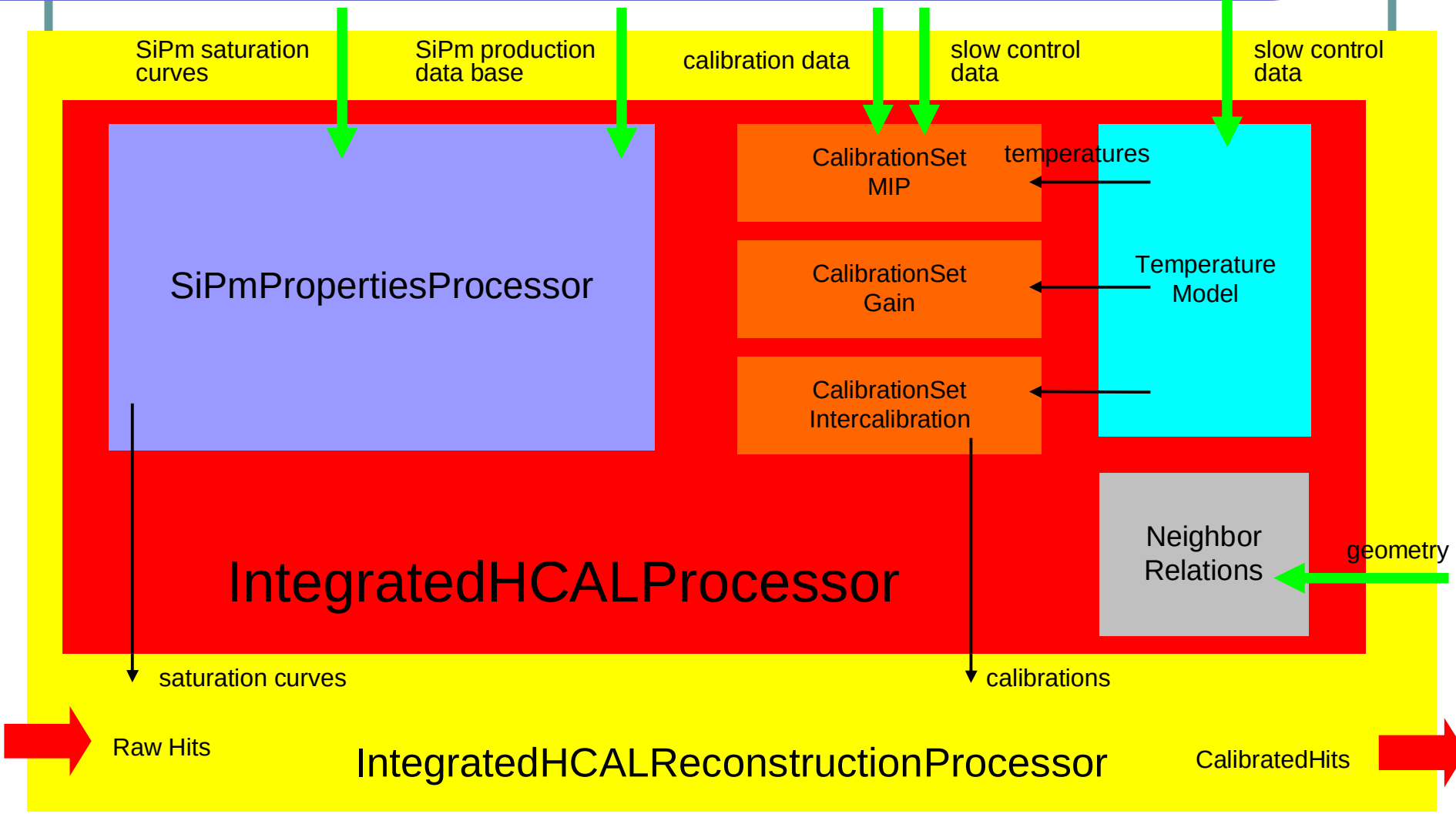
FastMappingMCProcessor

IntegratedHcalDigitizationProcessor

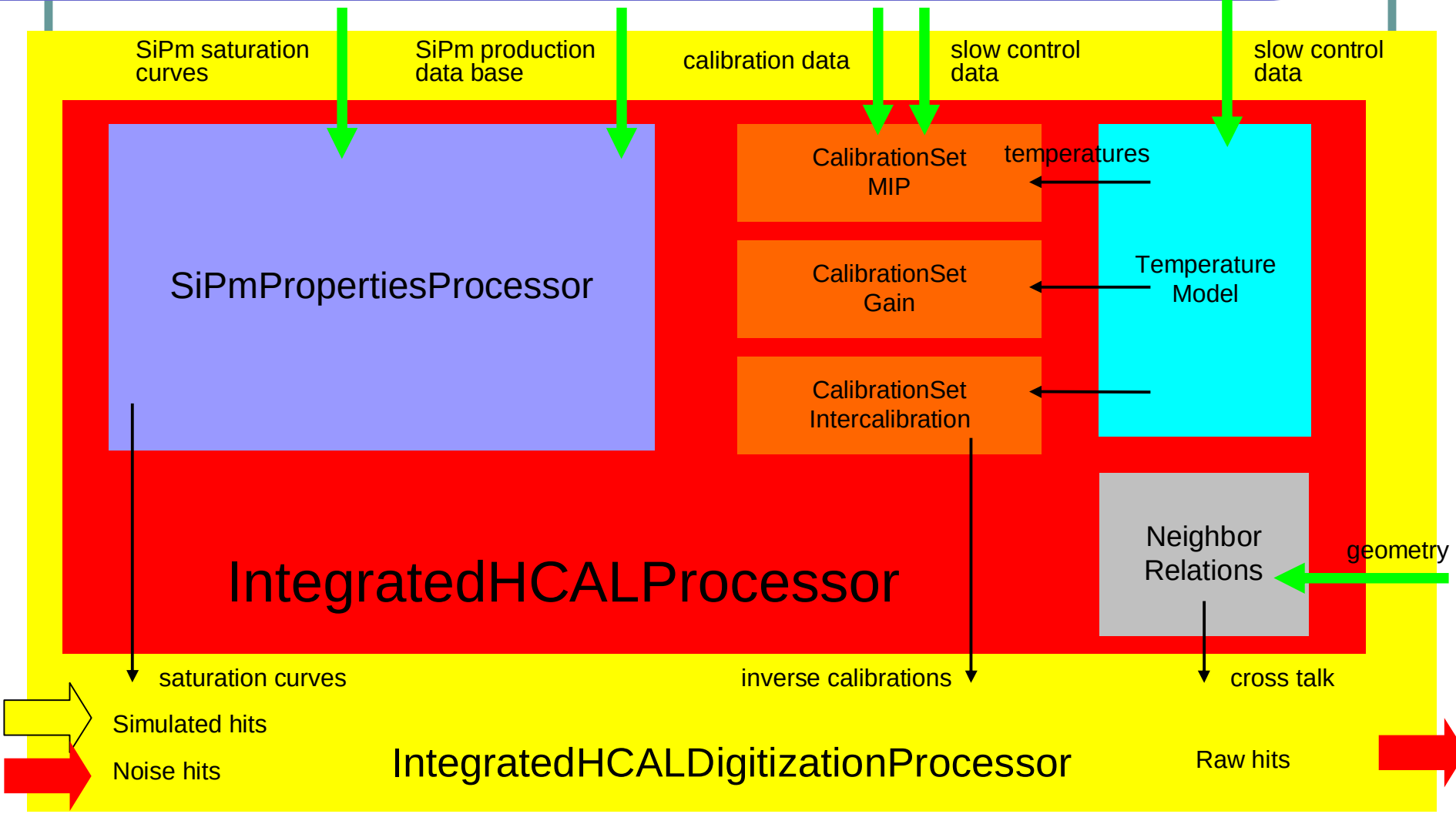
IntegratedHcalReconstructionProcessor

FastMappingIIProcessor

Integrated HCAL Processors



Integrated HCAL Processors (cont'd)



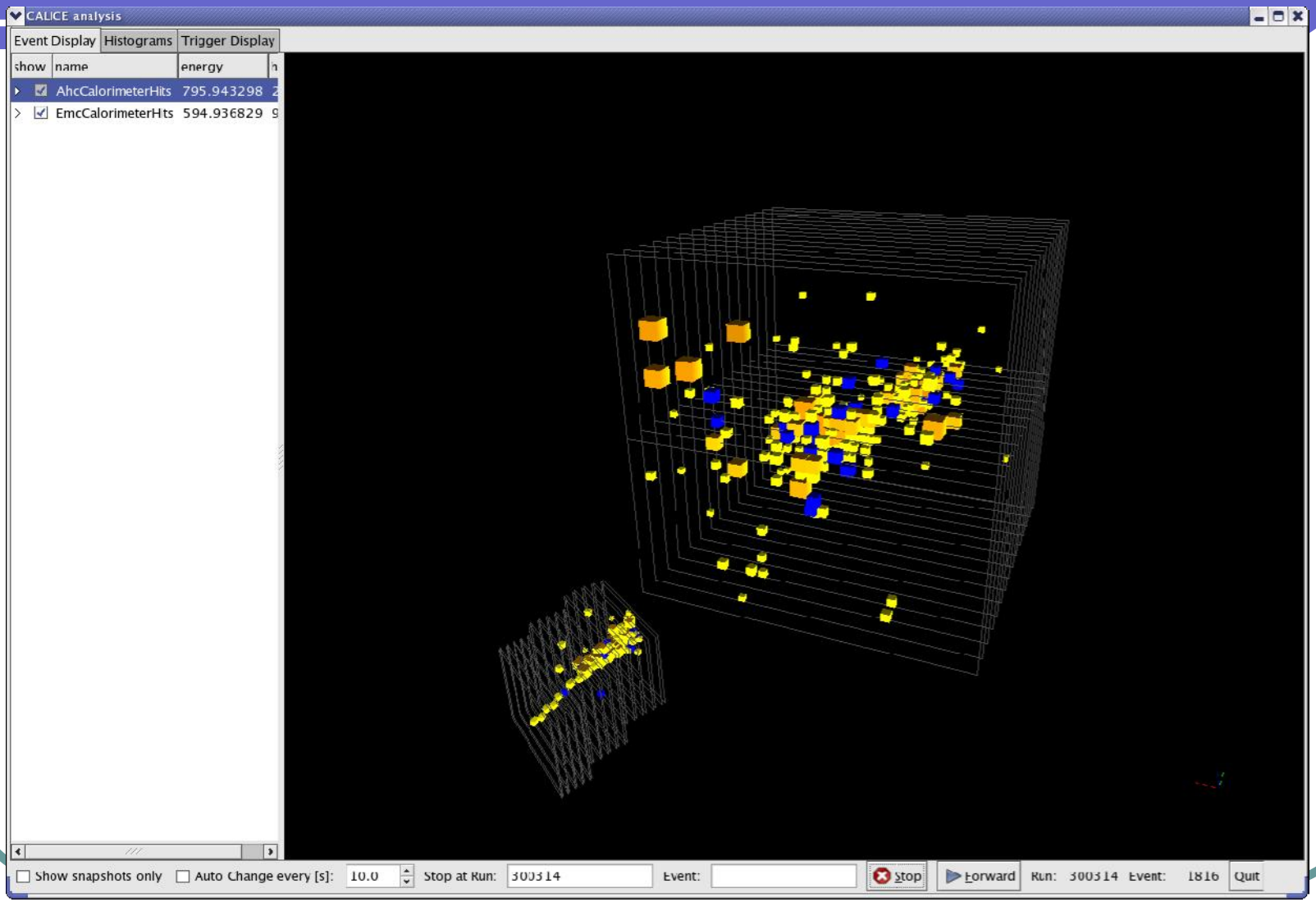
More classes for calibration

- CalibrationSet
  - Provides the module wise storage of calibration constants
  - Template of a certain calibration type
- TemperatureModel
  - Provides a temperature for every SiPm at every time
  - Depends on a arbitrary slow control collection, which contents are interpreted
  - Simple implementation exists
    - Direct assignment of cells and temperature sensors
    - No interpolation
    - Some fixes are implemented for buggy readout in 2006
- OverlayProcessor
  - Reads in collections from lcio noise files

Basic design of calibration constants

- Every calibration constant for a single cell is stored in an LCHcalCalibrationObject
  - The numbers stored in such an object are meaningless without the code to interpret them by definition
  - Calibration constants are only applied by usage of methods
- For every half module a collection of such objects exists in the data base
  - One can mix calibrations of different origin (MIP calibration from cosmics, from beam at CERN, etc.) for different modules
- LCHcalCalibrationObjects
  - can apply their calibration
  - can “unapply” their calibration (for digitization)
  - can be told about the temperature of their cell and can deliver results which depend on the temperature
  - can handle error propagation using the errors of the uncalibrated values, the uncertainty on the calibration constant from the data base and the uncertainty of the temperature
  - know about a criteria for zero suppression after they have been applied
  - can handle invalid calibration constants
- Extension needed for systematic studies
  - Introduce switch to change between the usage of nominal calibration constants/temperature and varied (up and down) calibration constants/temperature
  - Errors are already stored and propagated
  - Minor changes in LCHcalCalibrationObject, CalibrationSet and IntegratedHcalProcessor

After reconstruction



Outlook

- Minor things missing (mostly already prepared in the code)
  - Fill data base with rotated detector geometries, clean up data base
  - Cross check digitization
  - Include current HCAL reconstruction/digitization into official CALICE software and reconstruction
  - Usage of temperature dependent calibration constants in the official reconstruction
  - Data base based processor to cut out events from time ranges where magnets failed, etc. (store a single time dependent bit in the data base). From the experiences of my last shifts you need this one for sure to make any use of the data up to a 1-digit percent level.
- To be included in the next release of the HCAL software package
- And then finally: Forget about all the calibration stuff and do real physics (clustering, shower models, Higgs search, …)!
- Thanks for having a nice time in the CALICE collaboration!

backup

- One or more iterations for reconstruction? Extraction of pedestals in first run might be desirable, extraction of calibrations itself is anyhow done manually, no alignment done so far
- Variation of calibration constants according to data base, totally transparent to the analysis job, temperature can change often during run
- Variation of calibrations possible with extension, complete reconstruction should be rerun and is possible
- Data base folder names might depend on site, parameters shouldn’t
- MC events are digitized for a special run number (important to spread over complete run range)
- All known effects have been included into MC digitization (except addition of pedestals)
- Reconstruction is the same for data and MC, except for pedestals