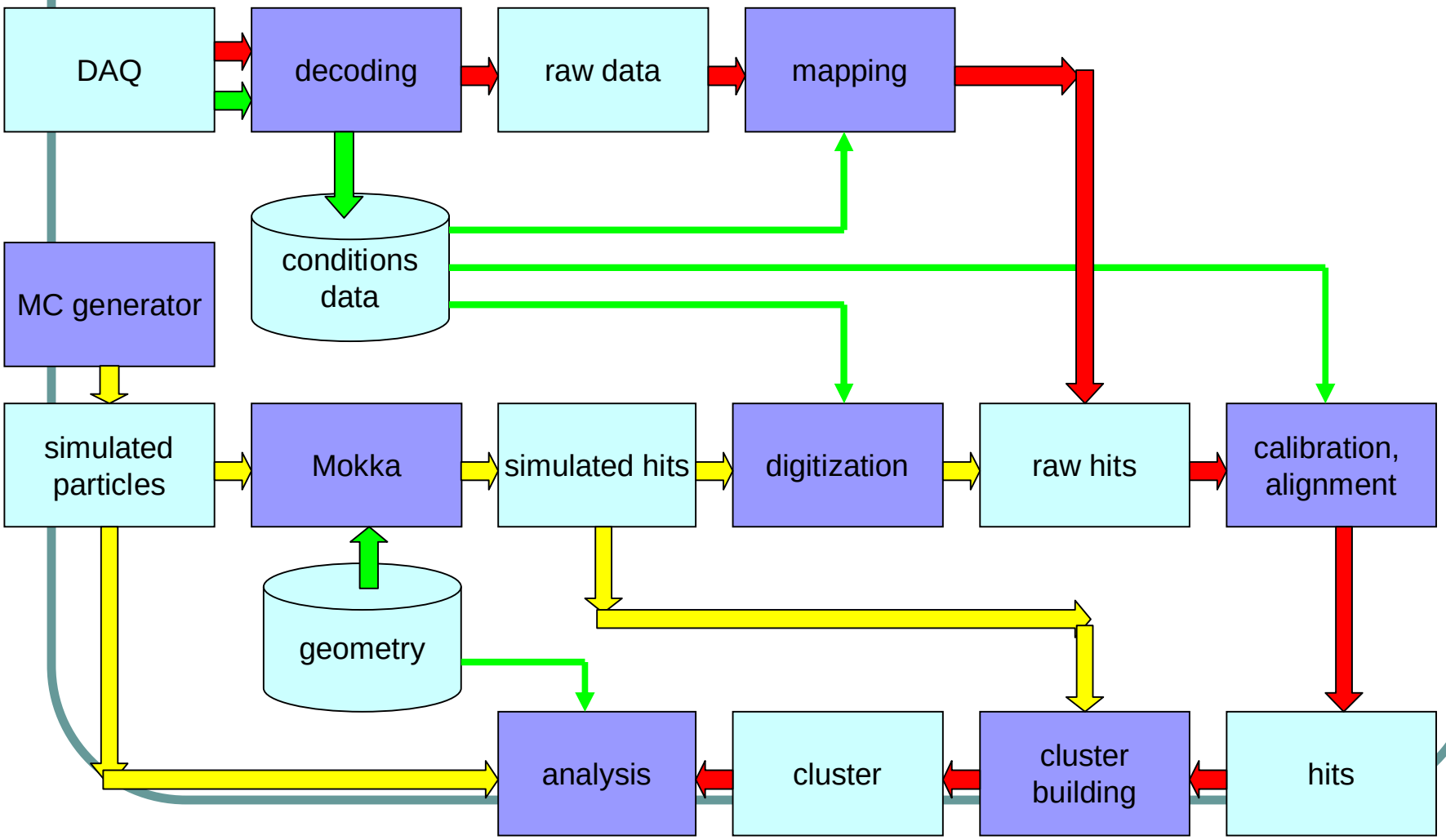
HCAL reconstruction software

Sebastian Schmidt, DESY

CALICE meeting, Hamburg

18-APR-07

CALICE data streams



Current release of HCAL software

- hcal-v00-01-18 available since early March
- Includes everything promised in the last meeting
  - Implements detector positions/alignments
    - if you have complaints about the current alignment/a better alignment, please contact Roman
  - Access to PIN diodes
  - Complete web documentation of all classes
  - Simplified installation routine: with one command you get the complete up-to-date setup
  - Conditions data base viewer included
  - Examples of how to explore the conditions data base
- And even more …

Current release of HCAL software

- Muon finder (Nicola)
- 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 …)
- SL4 ready
- Set of steerings used for the creation of HCAL conditions data available on the web
  - Successfully used to setup detector geometry/cable mapping for DESY test beam

Official CALICE reconstruction

- HCAL software included into official CALICE reconstruction software
- Test runs on the grid revealed some minor problems in the first version
  - Relative detector positions were wrong → fixed
  - Cell Index for HCAL wrongly implemented → fixed in the data base for /cd_calice/cern_beam_test (use that instead of /cd_calice/cern_beam/ if you are interested in Cell Indices and test it)
  - Missing modules in August lead to different reconstructions steerings for August and October data → to be fixed by dummy calibrations
- Now everything is understood and ok

Next release of HCAL software

- We have already quite some new things:
  - Processor which distributes simulated MC hits to hardware modules
    - necessary for digitization which depends on current hardware status
    - replaces mappingIProcessor for MC by mappingMCProcessor
  - SiPmPropertyProcessor (Erika) which
    - gives access to all SiPM quantities measured at ITEP, including saturation curve
    - and which knows about the positions of all the SiPMs in the modules
  - Interface to Vassillys DeepAnalysis, sampling fractions are calculated automatically from calorimeter geometry in the data base
  - First version of drift chamber reconstruction
  - Deletion of conditions data folders with the database viewer possible (Roman)
  - And: bug fixes
- I'm still collecting more things for the next official release, if you are interested in one of the topics mentioned above you can get the code immediately!

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 (trivial c&p)

Summary & outlook

- Software framework is growing (and getting more and more complete) as more and more people contribute
- The frameworks soft skills also improve (documentation/convenience tools/automatic installation)
- Next steps planned
  - Collect all existing digitization, make it aware of conditions data
  - Include tail catcher geometry and reconstruction
  - Extend standard calibration routines towards temperature dependencies (gain, MIP, …)