THE SIMULATION- AND ANALYSIS-FRAMEWORK FAIRROOT

ROOT Users Workshop, Saas Fee

Outline

- FairRoot
- Design Goals
- Features
- Time Based Simulation
- How to use it for Online Analysis
- Summary

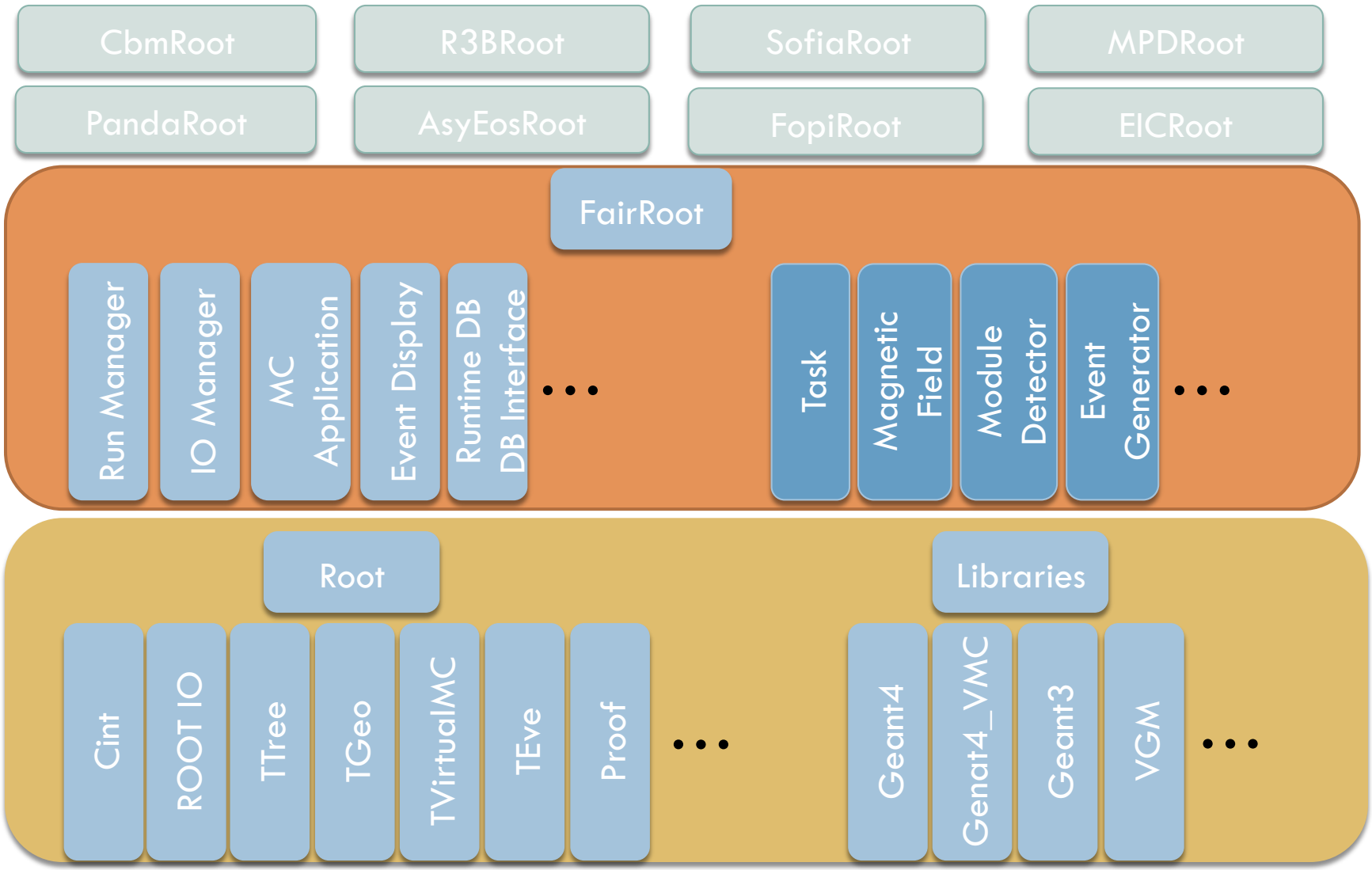
FairRoot

- Simulation-, Reconstruction-, and Analysis-Framework (not only) for the FAIR experiments
- Based on ROOT
- 2003 started as 2 person project for the CBM experiment
- 2013 ≈ 10 experiments use FairRoot as base for their developments
- Core team of 5 Developers (3.5 FTE)
- Many people contribute to make the project a success

Design Goals

- Avoid early decision about concrete simulation engine (Geant3 vs. Geant4)
  - Use different simulation engines (Geant3, Geant4, …) with the same user code (Virtual Monte Carlo)
- Reuse existing software and tools, use standards
  - Here ROOT comes into the game
- Code should run on all platforms
- Framework should be
  - Easy to install
  - Easy to use
  - Should allow fast development cycles
  - Flexible to easily change experimental setup
  - Extensible for new developments

Design


CbmRoot
R3BRoot
SofiaRoot
MPDRoot
PandaRoot
AsyEosRoot
FopiRoot
EICRoot
FairRoot
Run Manager
IO Manager
MC Application
Event Display
Runtime DB DB Interface
...
Task
Magnetic Field
Module Detector
Event Generator
...
Root
Cint
ROOT IO
TTree
TGeo
TVirtualMC
TEve
Proof
...
Libraries
Geant4
Genat4_VMC
Geant3
VGM
...

Features

- Easy to install
  - Provide package with all dependencies (ROOT, Geant3, Geant4, CMake, Boost, …) plus scripts for automatic installation
  - Use CMake as build system and CTest/CDash for automatic testing and QA
  - Works on Mac OSX and many Linux derivatives (Debian, Ubuntu, Suse, Fedora, Scientific Linux), probably on many more which are not tested
- Easy to use
  - In my opinion for sure
  - Unbiased answer: Ask the Users

Features

- Flexibility
  - Define run configuration at runtime
    - Use Root macros to define the experimental setup or the tasks for reconstruction/analysis
    - Use Root macros to set the configuration (Geant3, Geant4, …)
  - No executable
    - Use plug-in mechanism from Root to load libraries only when needed
  - No fixed simulation engine
    - Use different simulation engine (Geant3, Geant4, …) with the same user code (VMC)

Features

- Flexibility
  - No fixed navigation engine / geometry management
    - Use Root TGeoManager for geometry management
      - Geometry can be defined using different input formats
        - ASCII files in format inherited from HADES
        - Root files
        - Defined directly in the source code
    - Use TEve as base for general event display
    - Geometry is described once. Then it can be converted (VGM) to choose between different MC's and different navigations
      - G4 native geometry and navigation
      - G4 native geometry and Root navigation
      - …

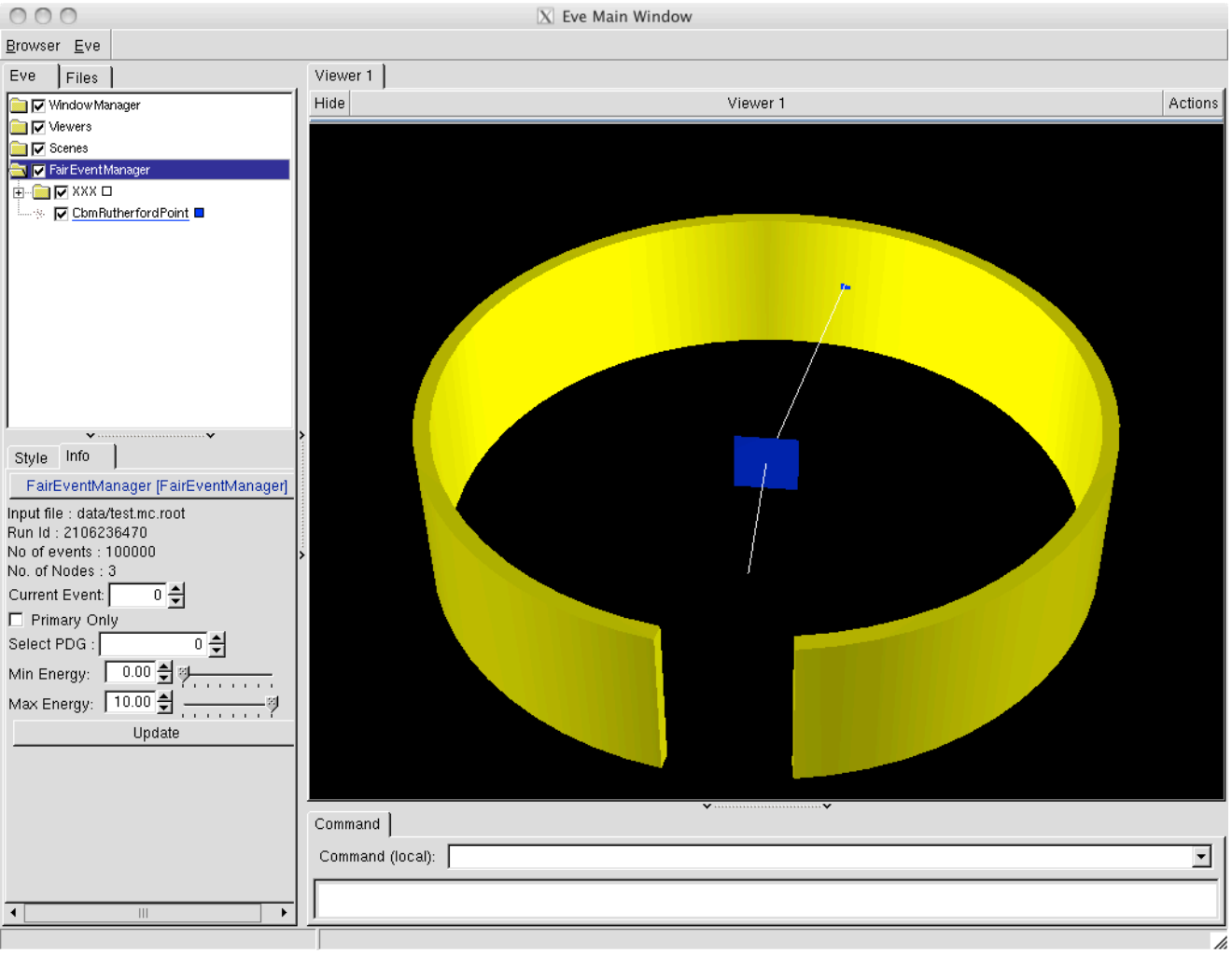
Features

- Flexibility
  - No fixed output structure
    - Store only the registered data classes to file
    - Use a dynamic event structure based on Root TFolder and TTree which is created automatically
    - Data output possible after each step
    - Simulation and reconstruction can be done in one go or in several steps
  - Parameter handling
    - Use runtime data base developed for the HADES experiment
    - Decouple parameter handling in FairRoot from parameter storage
    - runtime data base IO to/from
      - ASCII files
      - Root files
      - Database

Simulation Example: Rutherford Experiment

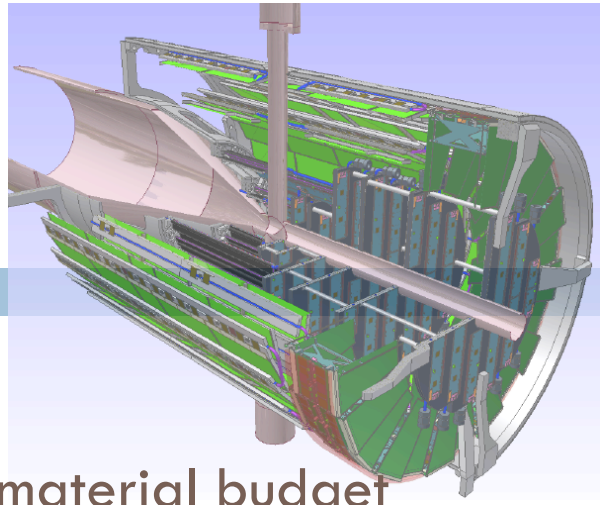
- Scattering of 5MeV alpha particles at a 2 μ m gold foil
- Unexpected large scattering angles observed
- Implementation using FairRoot needs
  - 600 lines of C++ source code created mostly automatically (copied from a template)
  - 60 lines of code for the build system
  - 200 lines of code for the steering macros
  - 70 lines of code for the geometry and media definition

Simulation Example: Rutherford Experiment

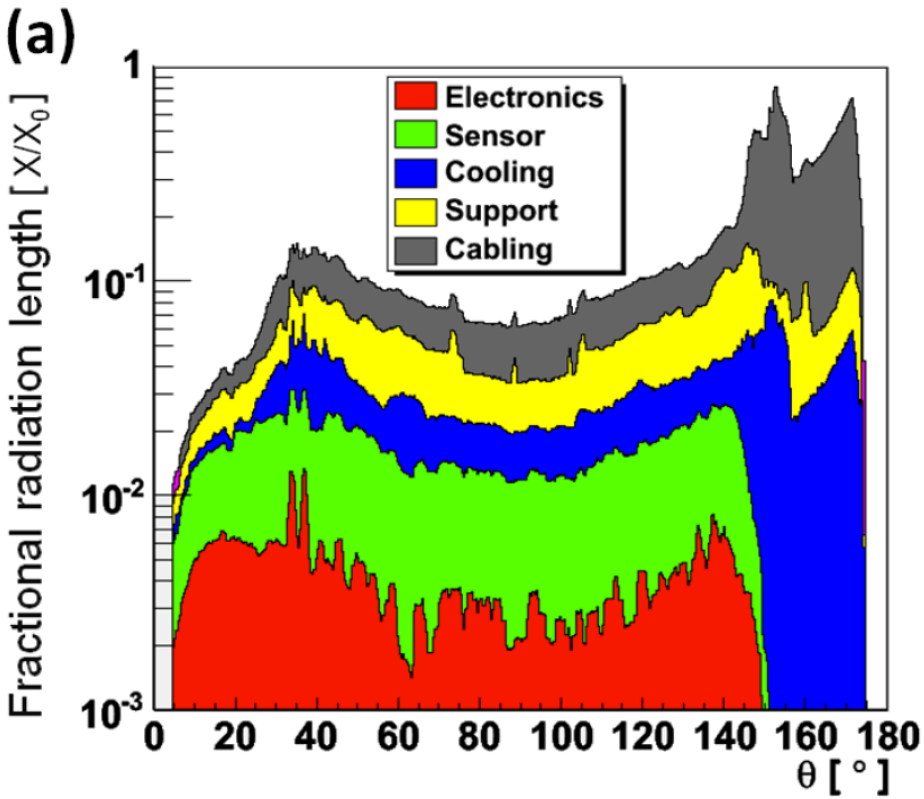


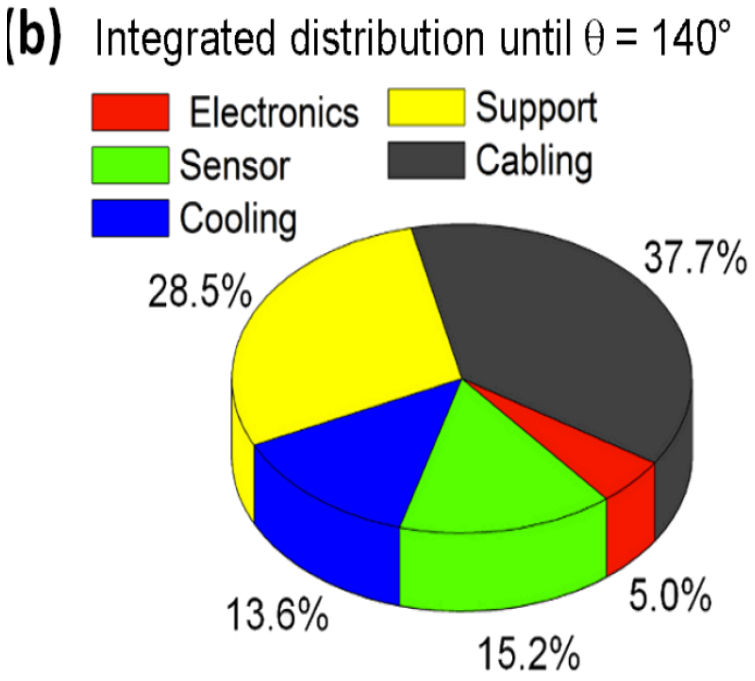
- Change experimental setup
- Change material properties
- Change simulation engine
- Change physical processes

Radiation length info FairRadLenManager



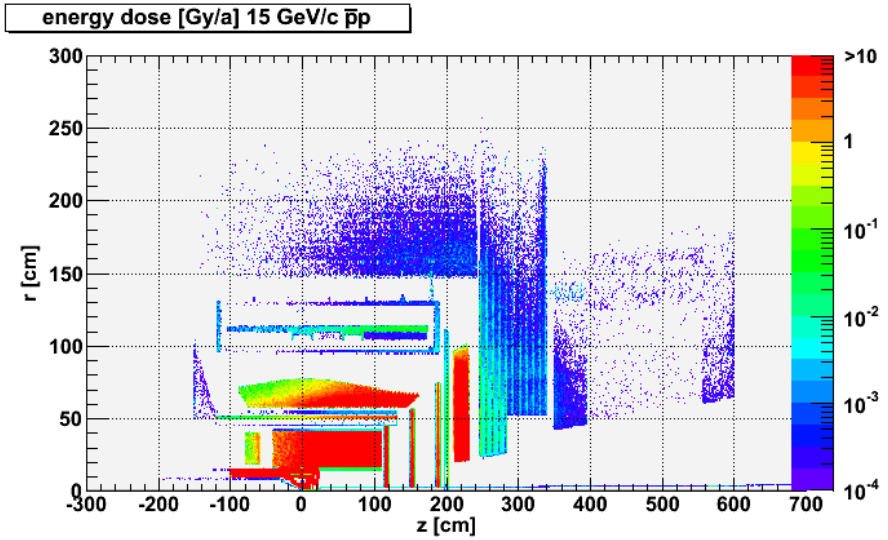
Example: Contributions of different
Functional parts of the PANDA MVD to the overall material budget





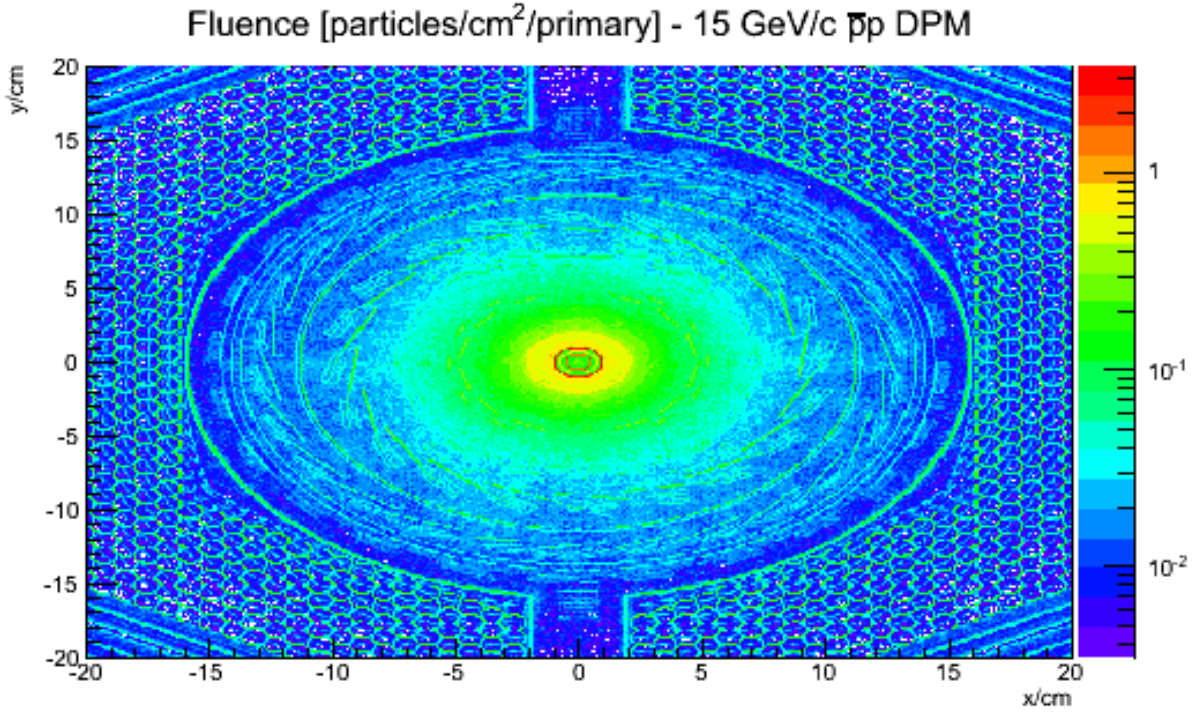
Dose studies FairRadMapManager

- What energy dose will be accumulated during a certain time of operation?
- Create all physical volumes with correct material assignment
- Run the simulation engine
- FairRadMapManager will sum up every deposited energy in each volume in the geometry



Dose studies
FairRadGridManager

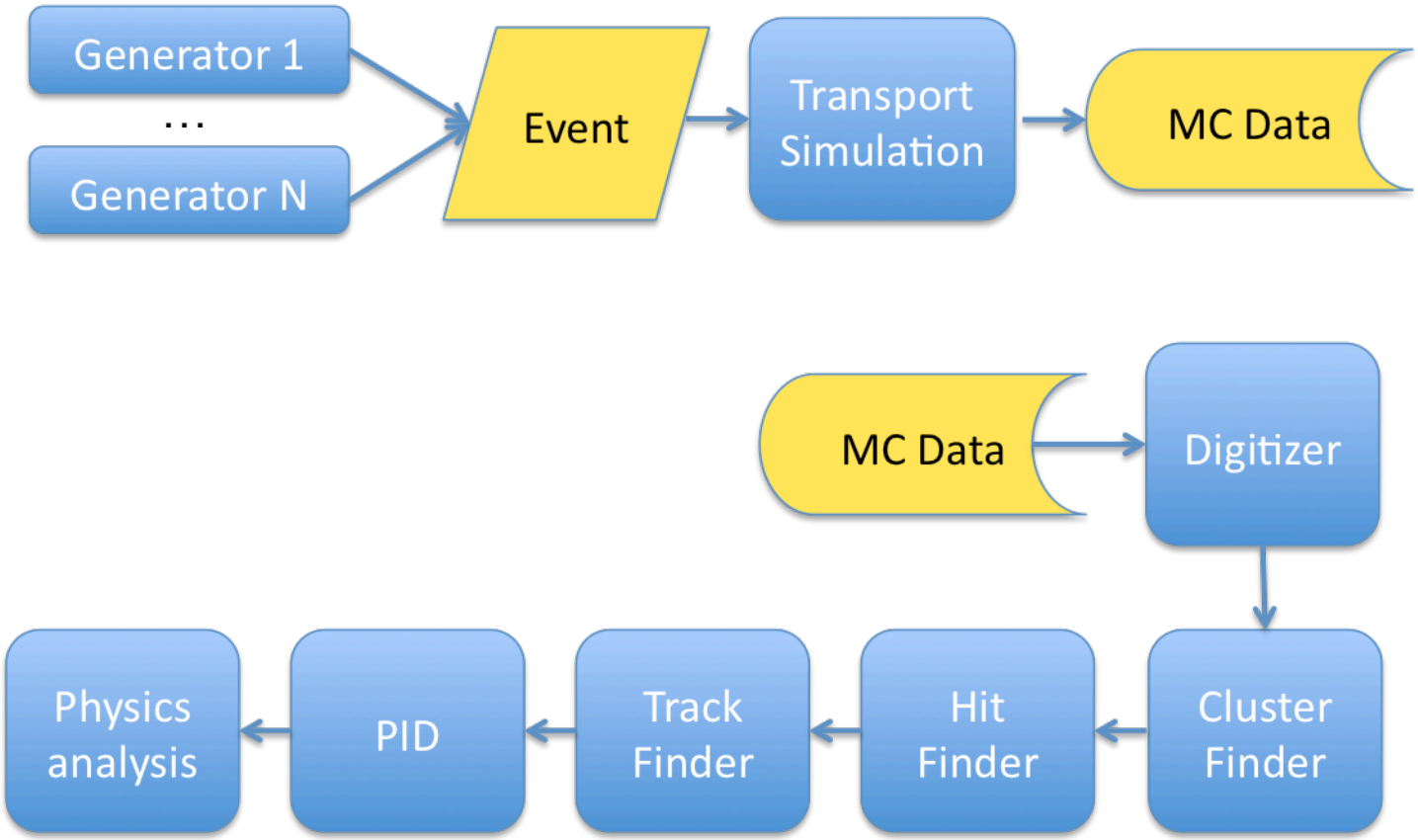
- Determine the particle fluency through a certain boundary (surface) and deduce a map. Knowing the volume and density of the object of interest and the specific energy loss doses can be estimated
- Surface does not interfere with real geometry
- Similar to scoring planes of Fluka



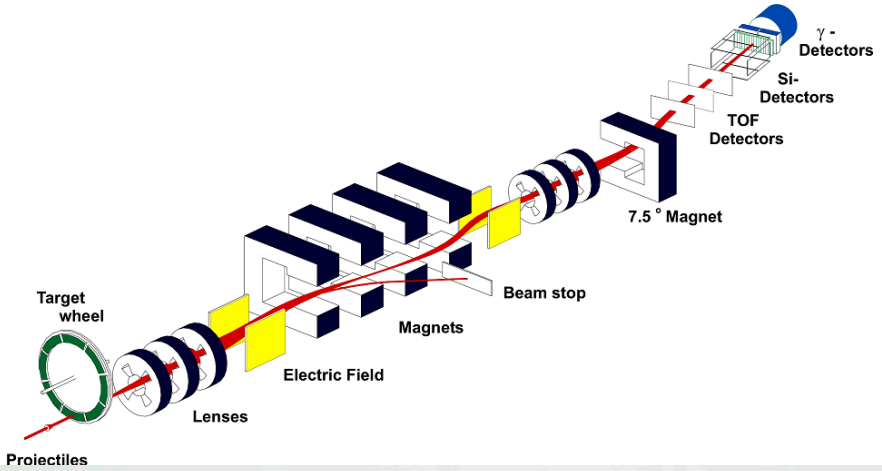
Reconstruction and Analysis

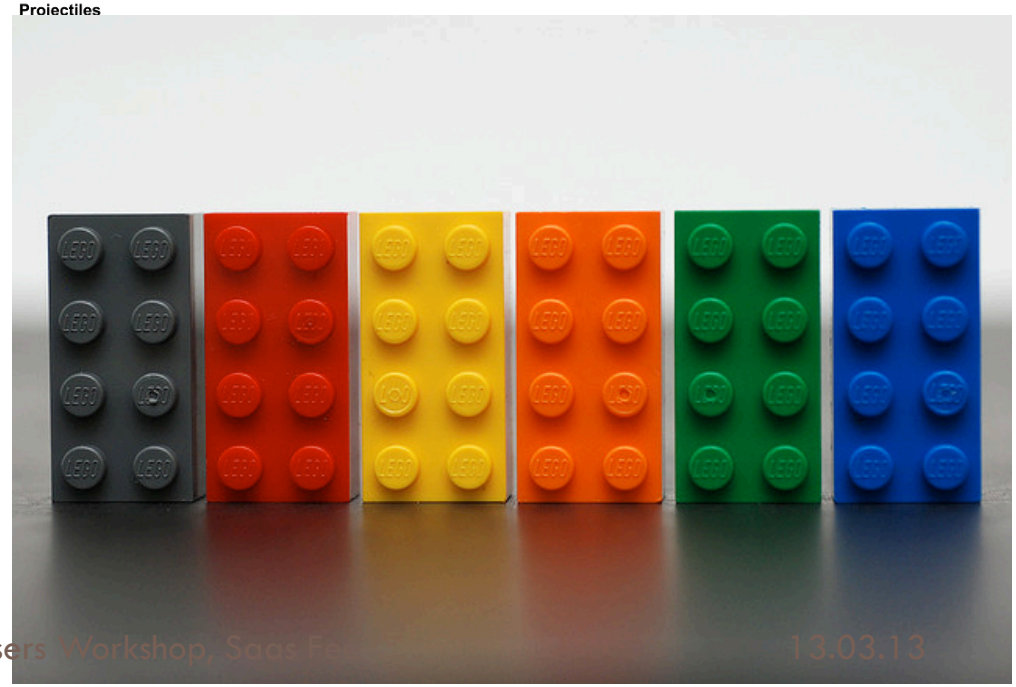
- Reconstruction and analysis are build out of tasks which are executed in a defined sequence
- Define needed input data and parameter containers
- Register output data at the IO Manager
- Do something useful for each event and convert input data to output data

Event based Workflow

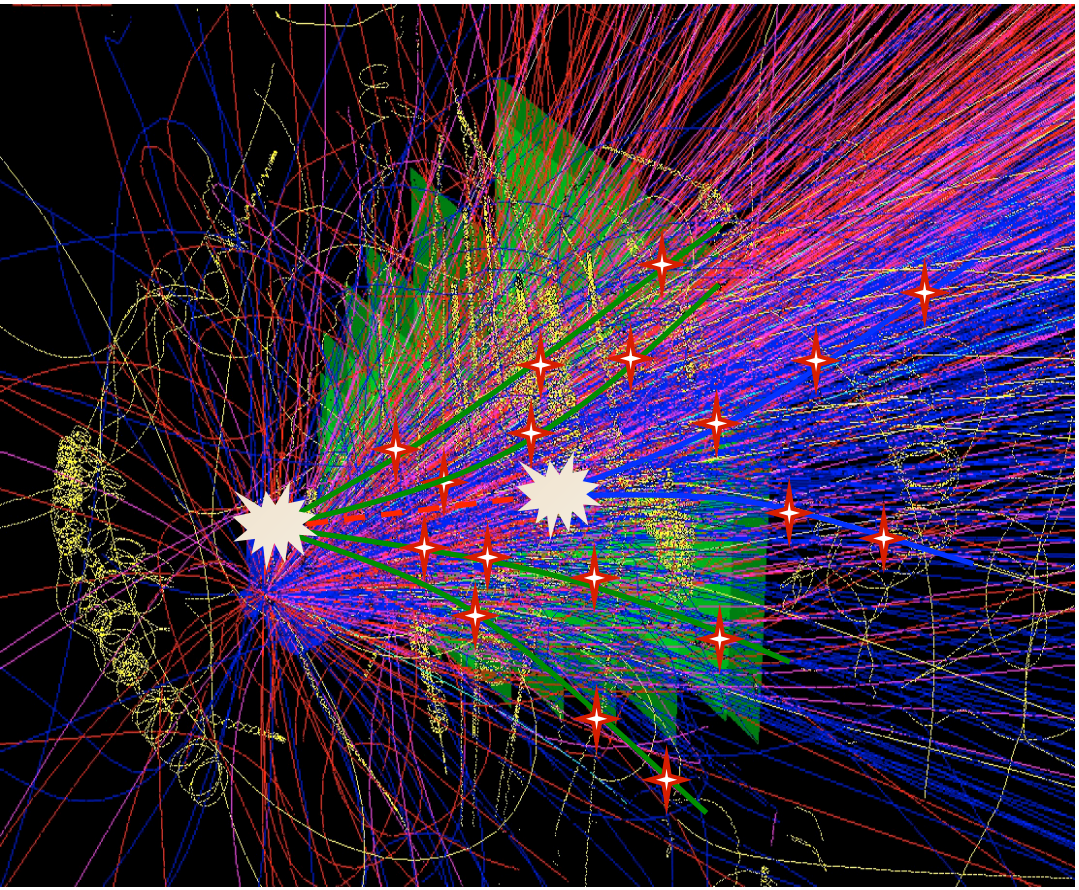


Triggered Experiment



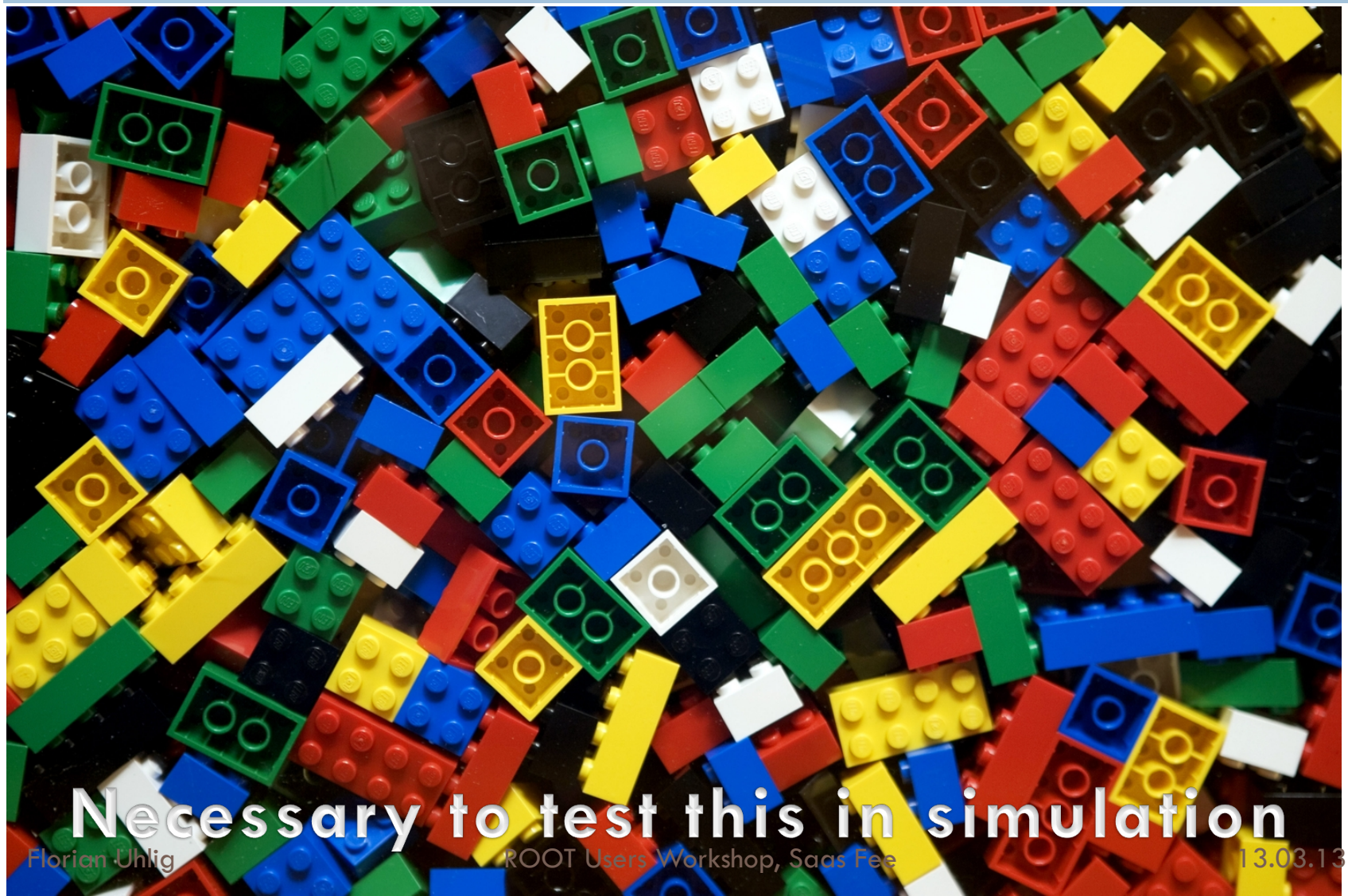


Problem: Self-triggered Experiment

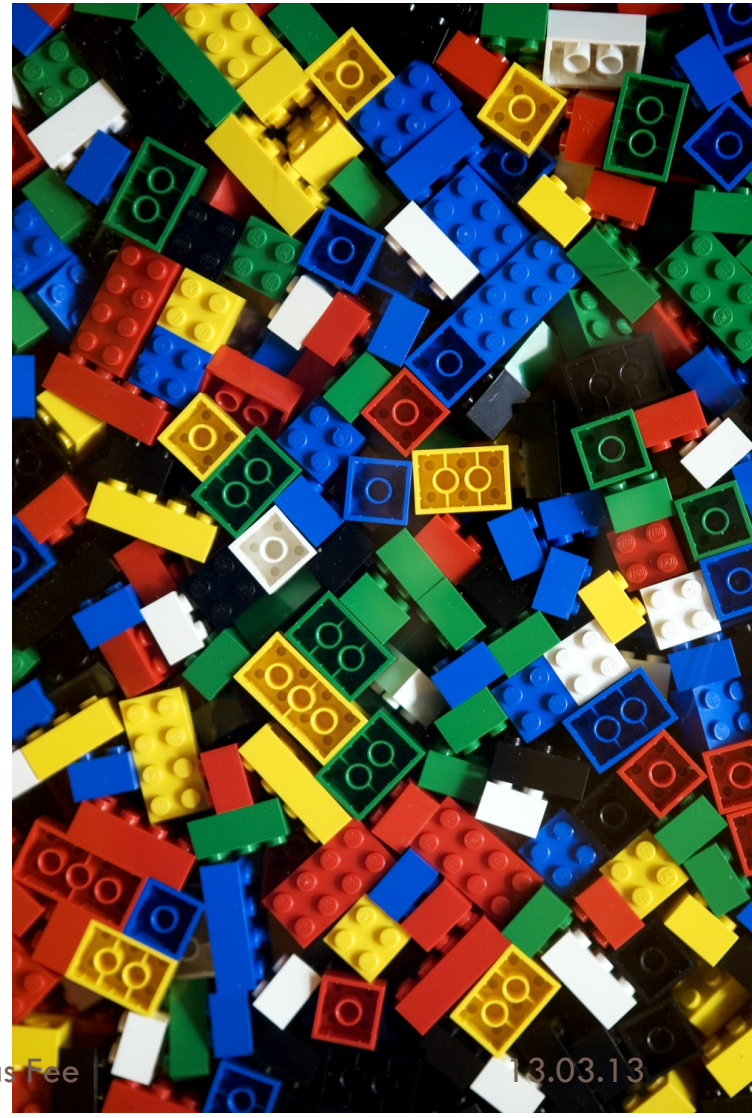


- Central events have up to 1000 charged particles inside acceptance
- Looking for rare probes require events rates up to 10^7 per second
- Complicated trigger signature
- Searching for secondary vertex requires reconstruction of a large part of the event
- Conventional hardware trigger not feasible: no dead time allowed
- Self-triggered autonomous front-ends pushing time-stamped data forwards to DAQ

The Challenge

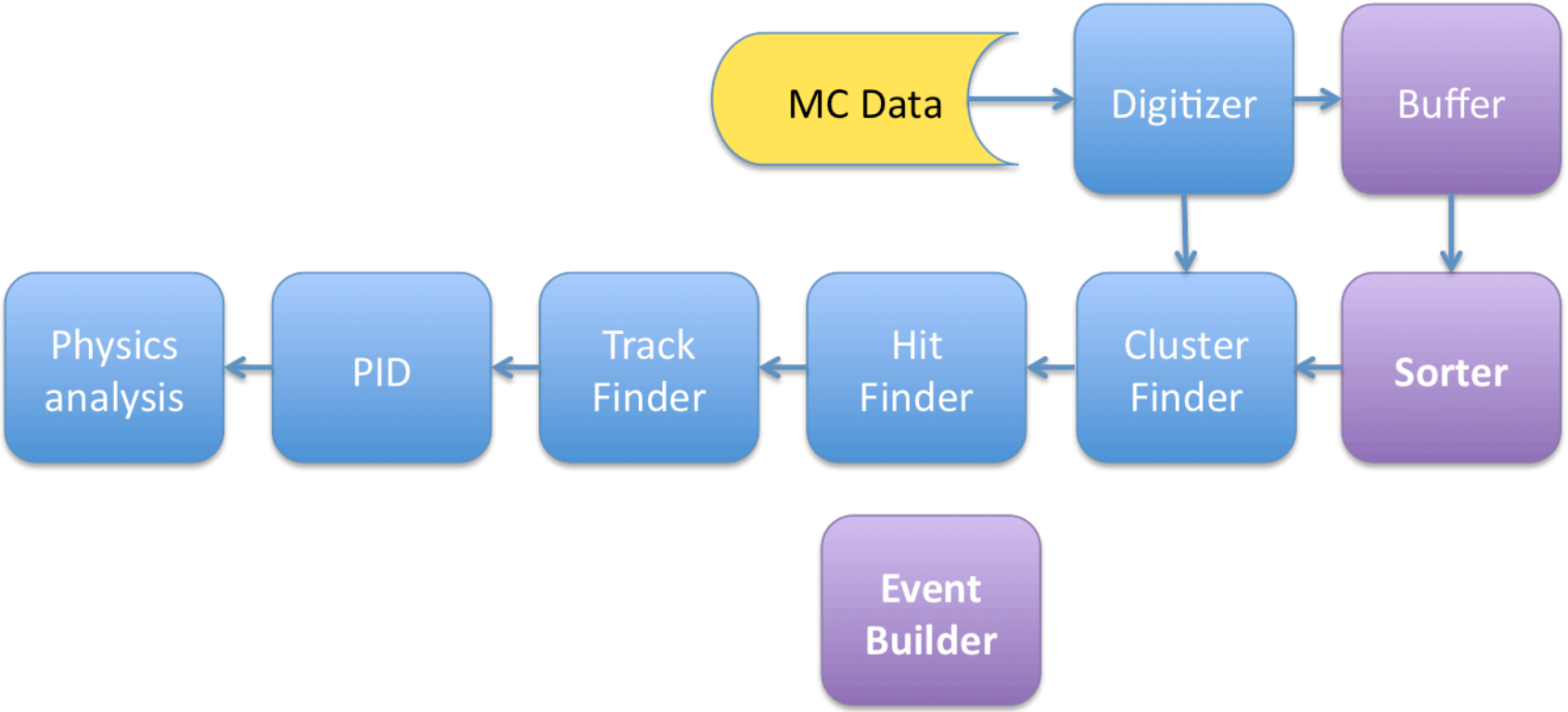


Task for FairRoot

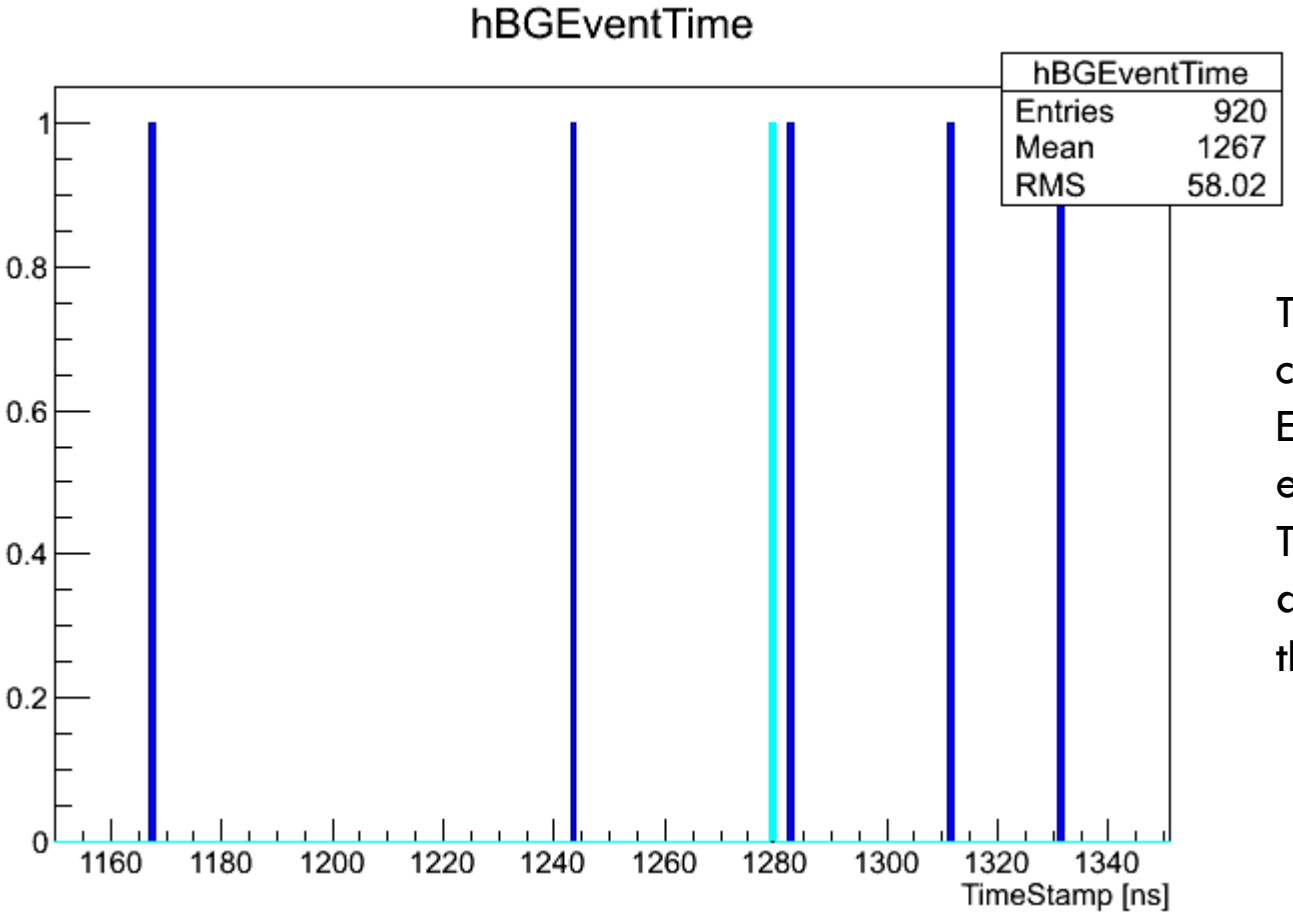


Time based simulation

- Provide functionality for the tasks “event mixing” in the digitization stage and “time sorting” in the reconstruction stage



Event Time

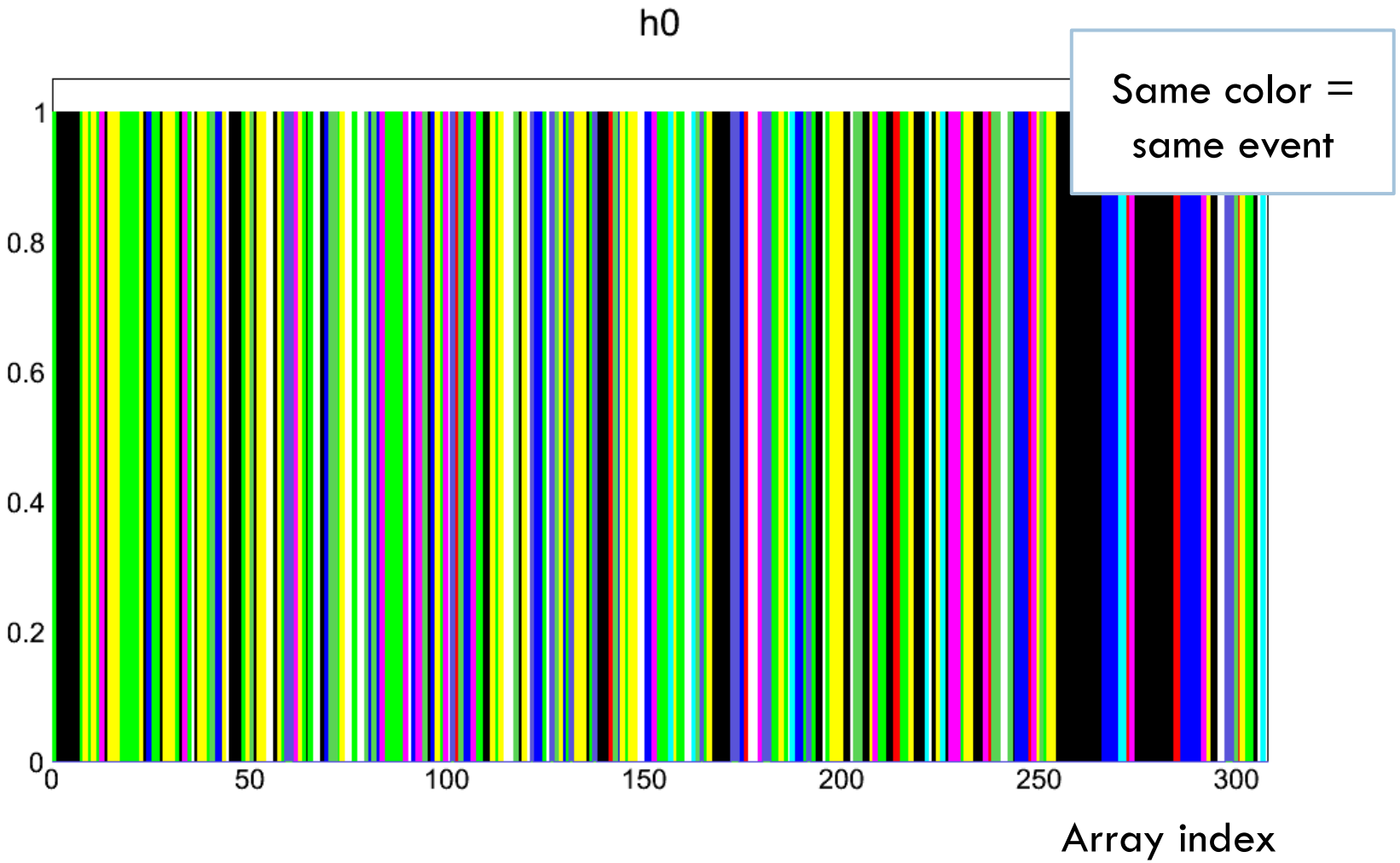


The absolute event time is calculated by the framework
Experiments define functions for event time calculation
Time of detector digi is this absolute time + the time inside the MC event

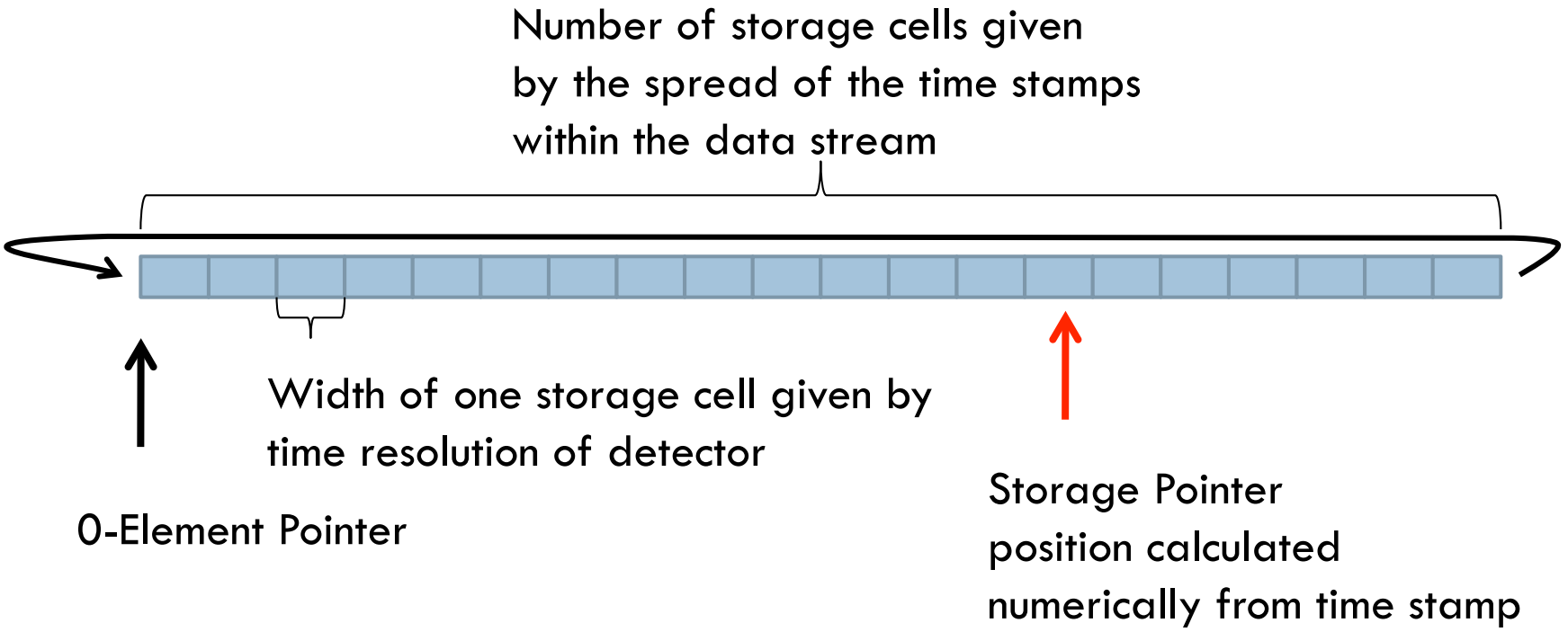
FairWriteoutBuffer

- Base class to store detector data (digis) between events
- Buffer stores data together with absolute time until this data is active and can be influenced by later events
  - This time is detector dependent and is defined individually
- If the same detector element is hit at a later time the data can be/is modified
  - Modifications are detector and electronics dependent
- Result is a randomized data stream which is stored in a TClonesArray which should mimic be the input to the DAQ in the real experiment

Randomized Digi Data

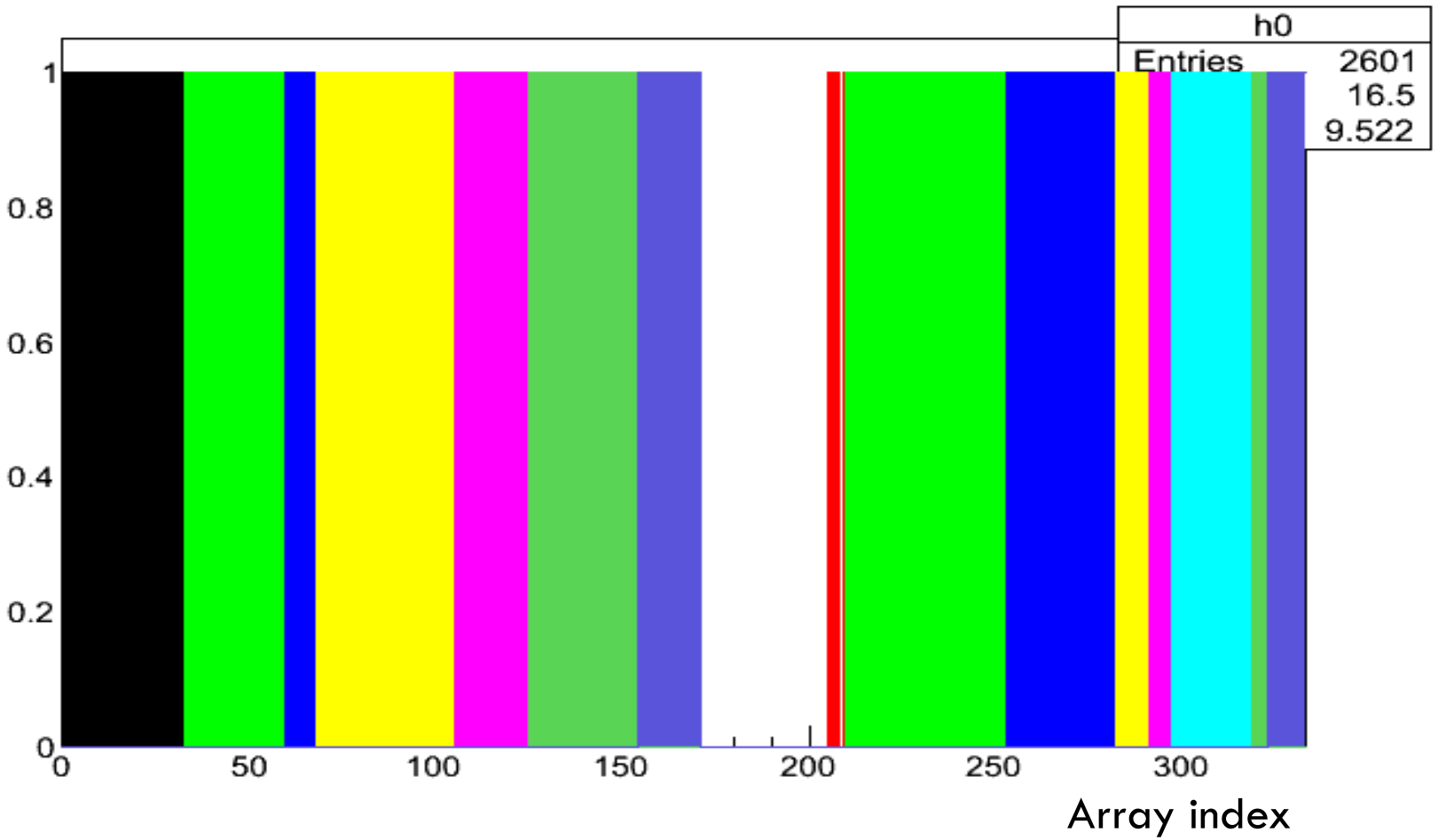


Sorter – Technical Implementation

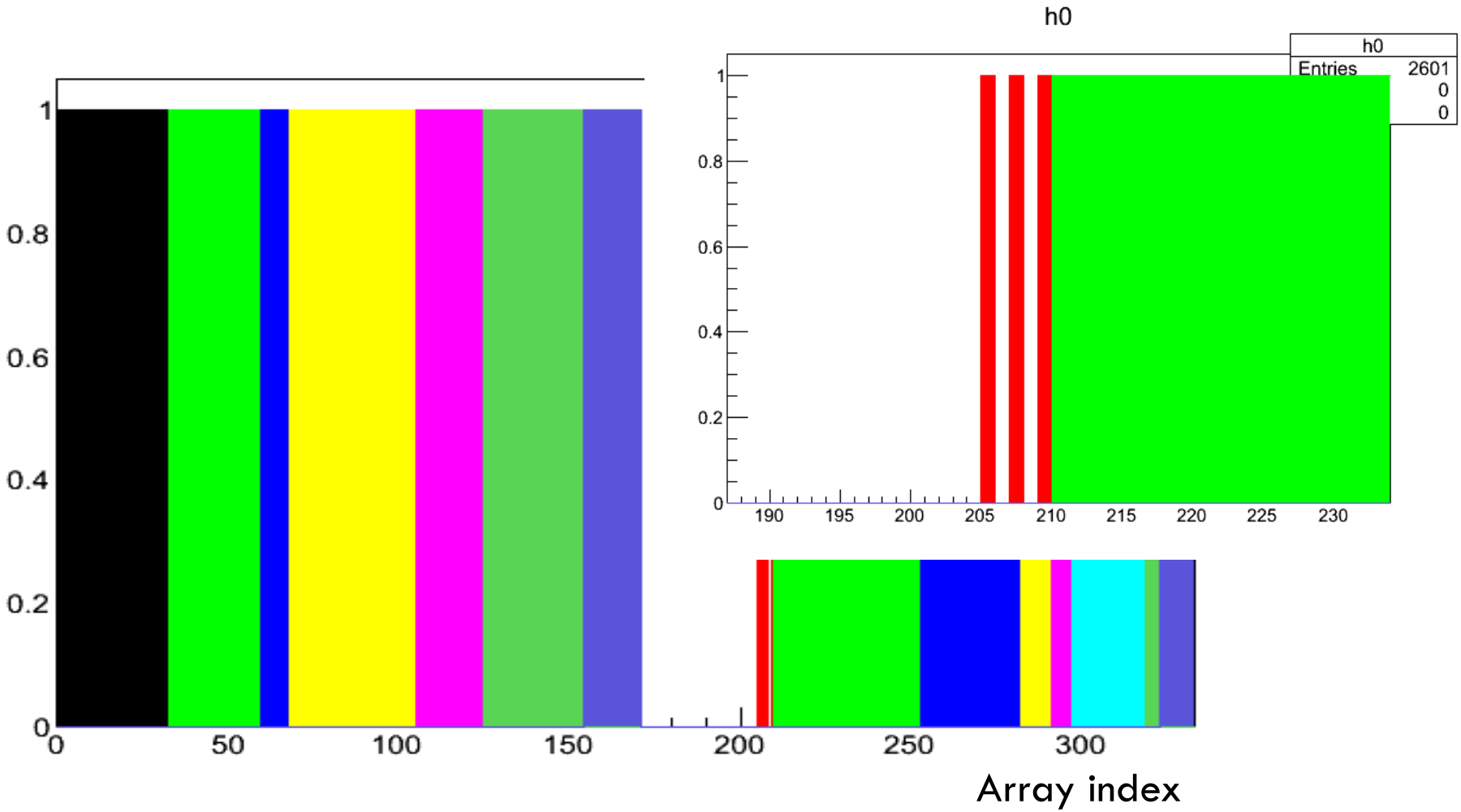


If a storage position is calculated which would override old data, the old data is saved to disk and the storage cell is freed

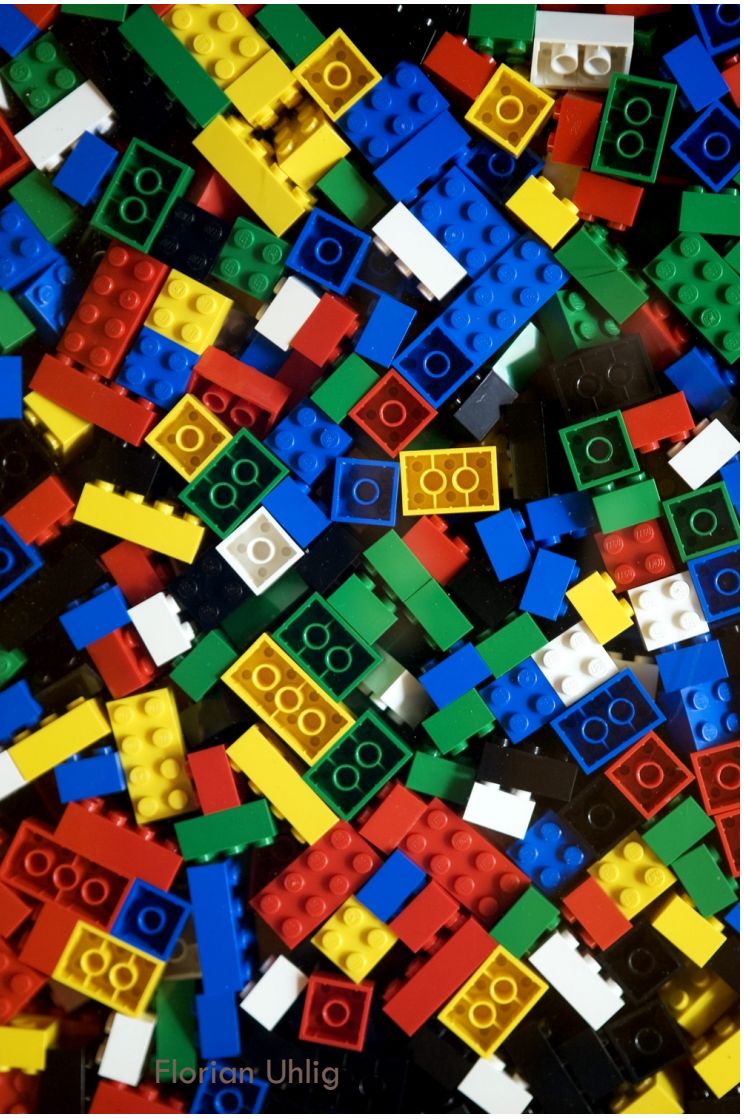
Sorted Digi Data



Sorted Digi Data



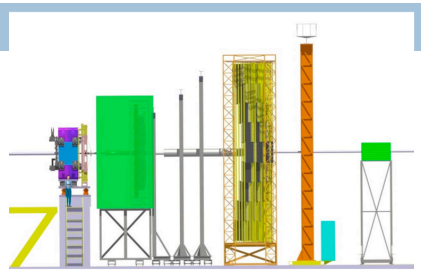
Task for Experiments Event Builder



Read back the Data Eventbuilder

- Read data from IO Manager using different functions
- Different algorithms already available
  - Read data up to a given absolute time
  - Read data in a given time window
  - Read data until next time gap of certain size
- Other algorithms can be (easily) implemented if needed

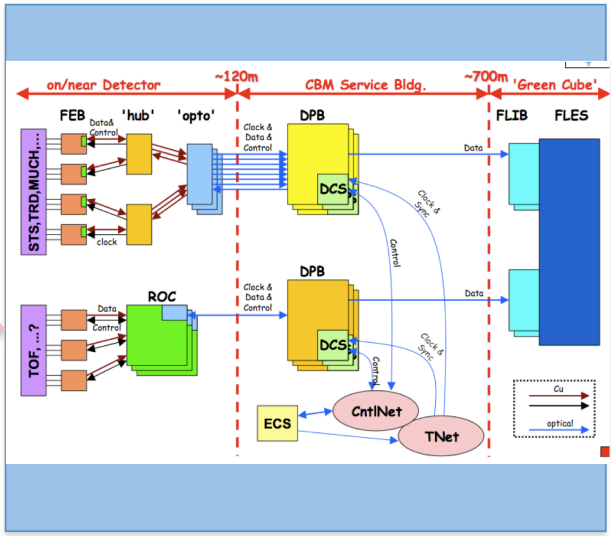
Next Problem: Do it online



CBM FEE

Online Computing Farm
~ 60.000 CPU Cores

1 GB/s – 1 TB/s



1 GB/s

Mass Storage

- How to manage the data flow on such a huge cluster?
- How to recover single/multiple processes?
- How to monitor it?

Data transfer network for FairRoot

- FairRoot must be extended to support the continuous pipeline-processing scenario of the online analysis
- Tasks have to be put on different compute nodes
- Transport data using message queuing technology
- Scheme would also allow to enable concurrency in FairRoot for offline analysis
- The long term plan is to have the same framework for online and offline

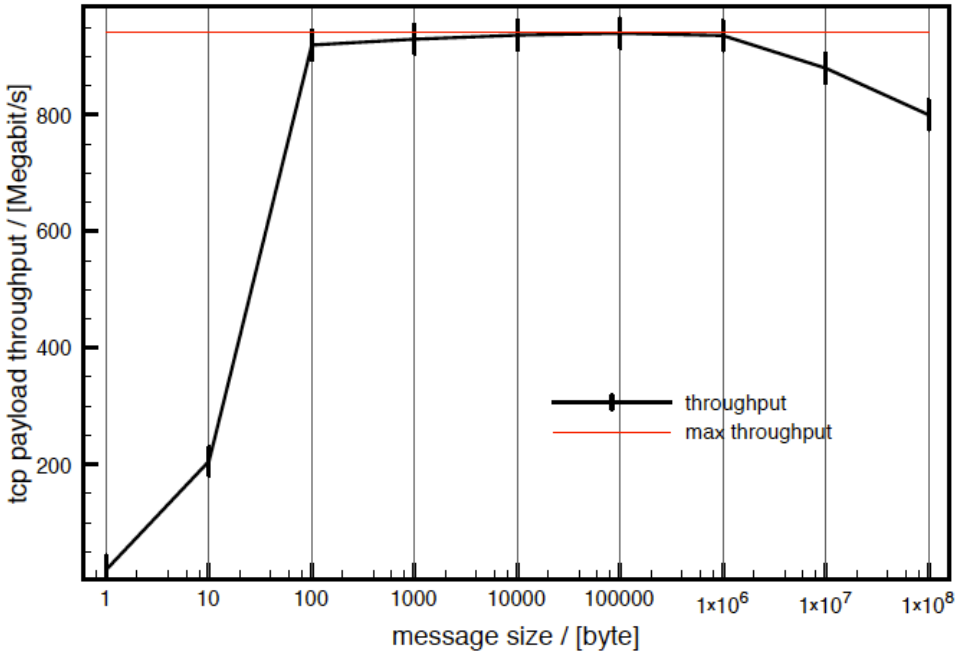
ØMQ: a library for message passing

- Looks like a networking library but acts like a concurrency framework
- Carry atomic messages across various transports like in-process, inter-process, TCP, and multicast
- Connect N-to-N with patterns like fan-out, publish-subscribe, task distribution, and request-reply
- Allows scalable multicore applications, build as asynchronous message-processing tasks
- Runs on most operating systems
- API for many programing languages
- Open Source

Test setup and results

Four identical nodes were connected to a GigabitEthernet switch for testing

One node has a large set of time sorted digis in memory and sends this data to a receiver which dumps the data



TCP throughput of 117.6MB/s was measured which is very close to the theoretical limit of 117.7 MB/s for the TCP/IPv4/GigabitEthernet stack.

The throughput for the named pipe transport between two devices on one node has been measured around 1.7 GB/s

Summary

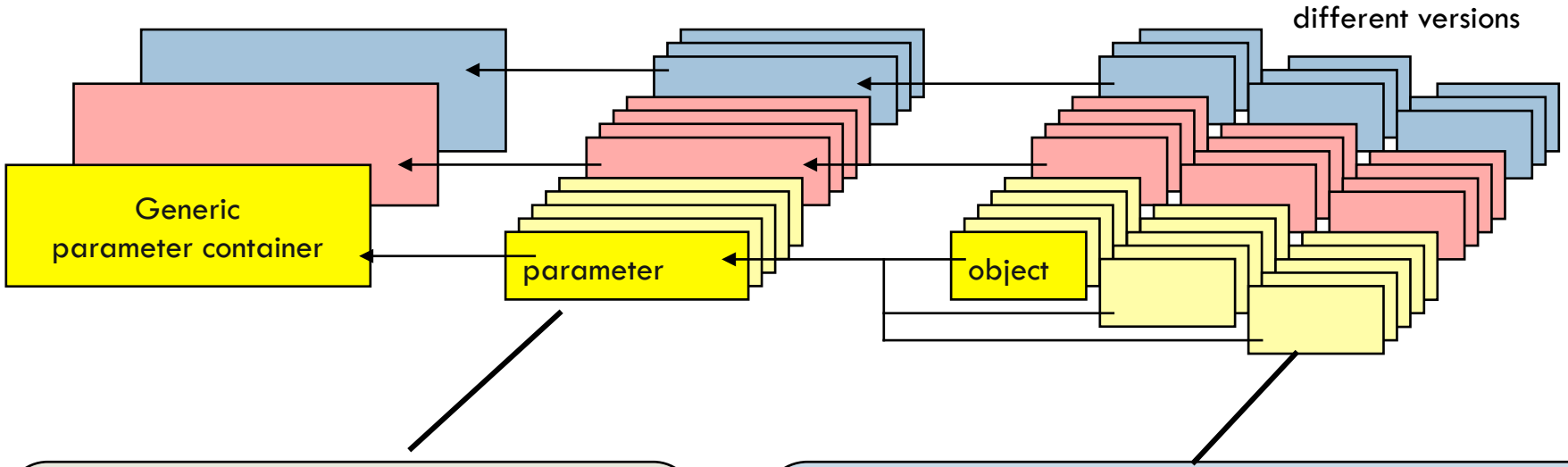
- Hope I could show you that FairRoot
  - is flexible
  - is easy to use
  - is easy to extend
- Special tools to do dose studies
- Tools for time based simulation are implemented
  - Calculation of event time
  - Mixing of events by automatic buffering and write out when needed
  - Fast sorting of data
  - Several event builder functions

Summary

- Many more topics not showed at all
  - Proof integration
  - Database connectivity
  - GPU usage inside of FairRoot
  - Build and test system
  - …
- Resources
  - Webpage: http://fairroot.gsi.de
  - Forum: http://forum.gsi.de
  - Test Dashboard: http://cdash.gsi.de/CDash

Backup Slides

Parameter IO

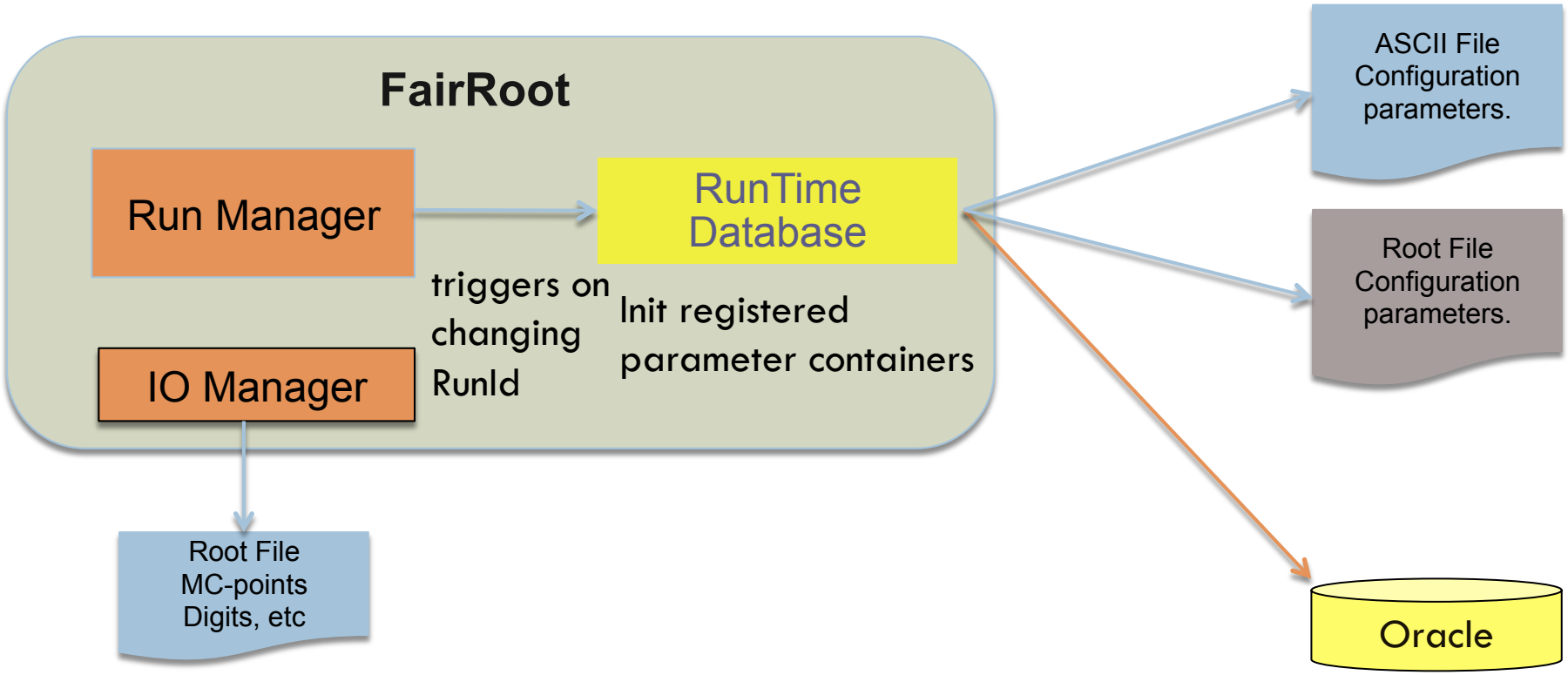


name
type: Int_t, Float_t, Double_t, Char_t, Text_t, UChar_t, class type
value(s)

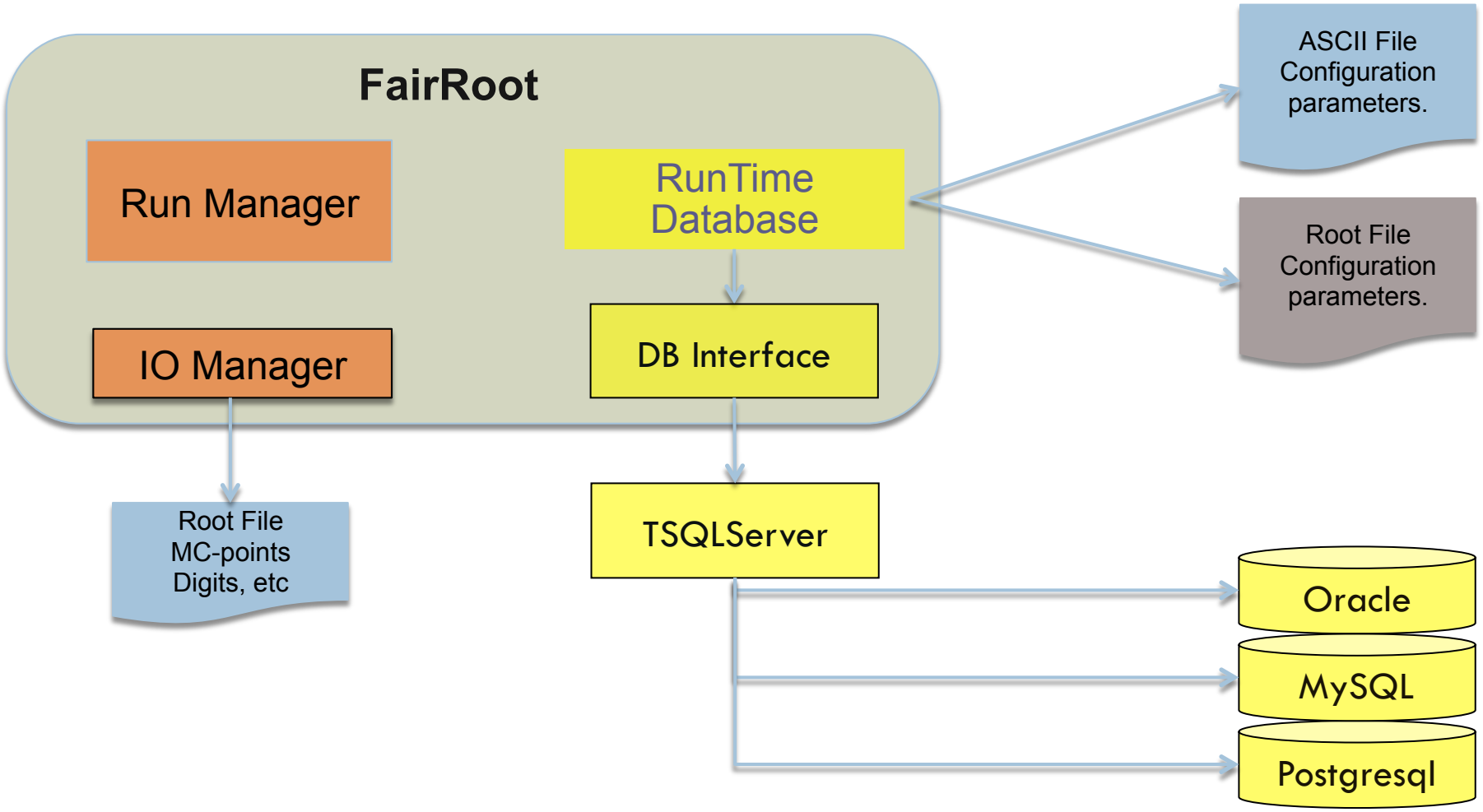
any class derived from TObject
decoded in the analysis interface by
ROOT streamer

stored as byte array (RAW or BLOB)
number of values (single value or array)
class version
streamer info, root version for ROOT classes
own version management

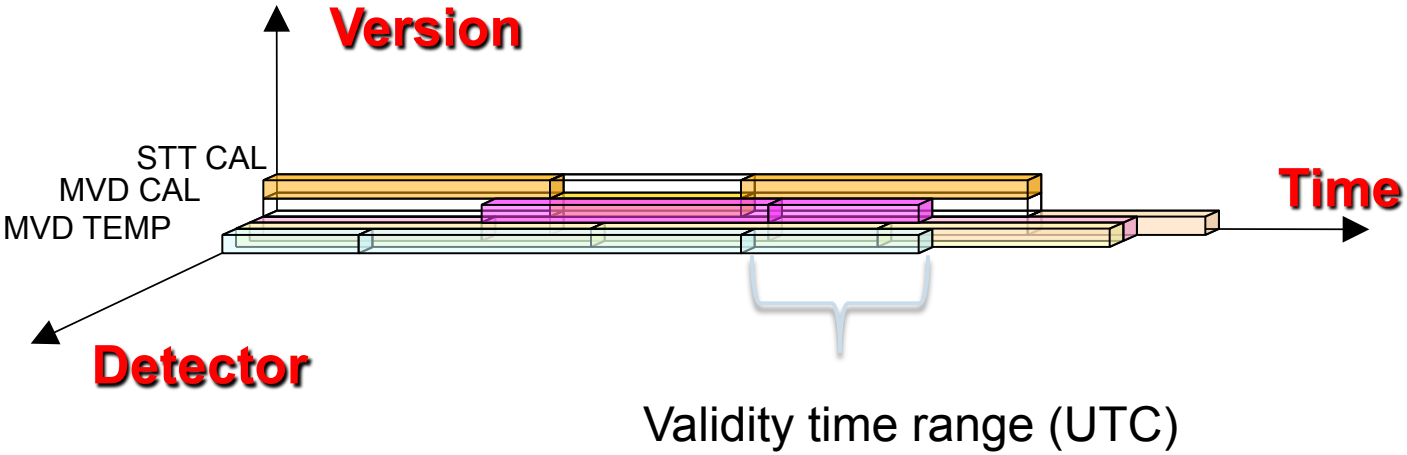
FairRoot DB


FairRoot
Run Manager
RunTime Database
triggers on changing RunId
Init registered parameter containers
IO Manager
Root File MC-points Digits, etc
ASCII File Configuration parameters.
Root File Configuration parameters.
Oracle

FairRoot DB extended



Version management



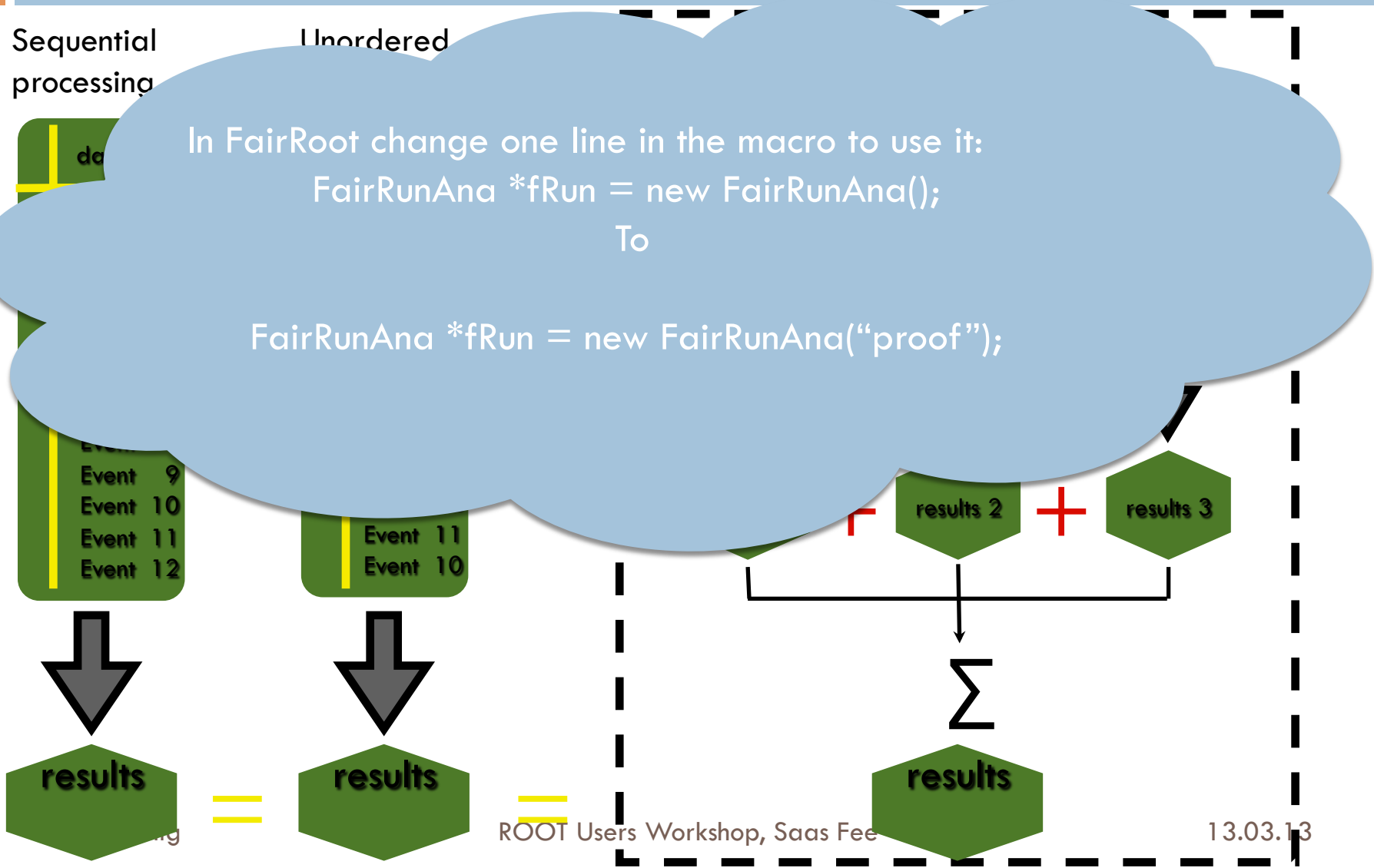
The Query process

1. Context (Timestamp,Detector,Version) is the primary key
2. Context converted to unique SeqNo
3. SeqNo used as keys to access all rows in main table
4. System gives user access of all such rows

Proof in FairRoot

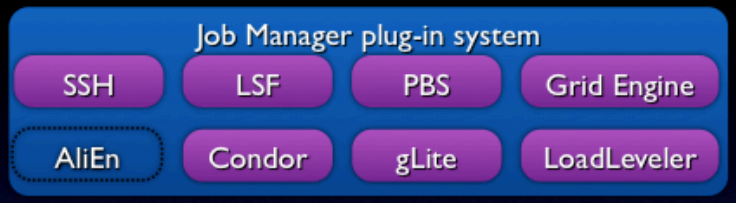
- PROOF - Parallel ROOt Facility.
- It allows parallel processing of large amount of data. The output results can be directly visualized (e.g. the output histogram can be drawn at the end of the proof session).
- The data which you process with PROOF can reside on your computer, PROOF cluster disks or grid.
- The usage of PROOF is transparent: you use the same code you are running locally on your computer.
- No special installation of PROOF software is necessary to execute your code: PROOF is included in ROOT distribution.
- Proof runs on computing clusters as well as on your local many core computer

Trivial parallelism



Proof on Demand

Different job managers



PoD is shipped with a number of plug-ins, which cover all major RMSs, such as local cluster systems and Grid.

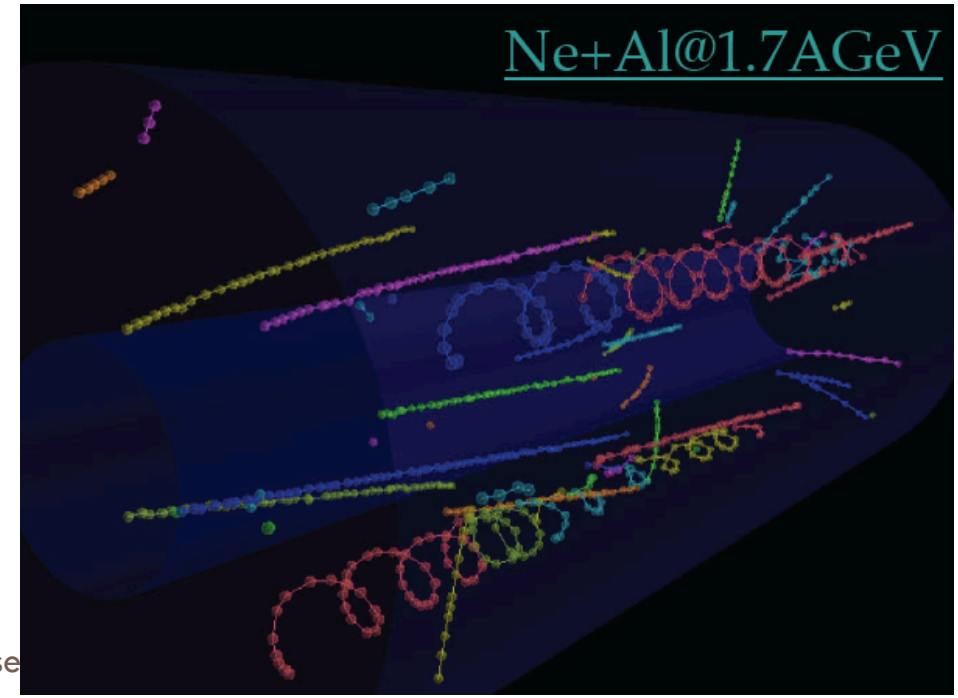
If you don't have any RMS, then the SSH plug-in can be used.
The SSH plug-in is also used to setup PROOF clusters on Clouds.

GPU support in FairRoot

- CUDA is fully integrated into the FairRoot build system
- CMake creates shared libraries for cuda part
- FairCuda is a class which wraps CUDA implemented functions so that they can be used directly from ROOT CINT or compiled code

FairRoot for real data

- FairRoot was designed from the beginning to combine simulation and analysis in one tool.
- Using the same internal structure the user can compare easily at any time/level the real data with the simulation



Reconstructed Beam EVENT

The large GEM-TPC Prototype
L. Fabbietti for the GEM-TPC Collaboration