IVCroot
the Software Framework of the 4th Concept

Software Guidelines

The Framework

General Architecture

Data Model

Montecarlo

Preliminary results

Corrado Gatto INFN-Lecce

lcws06 3/13/2006

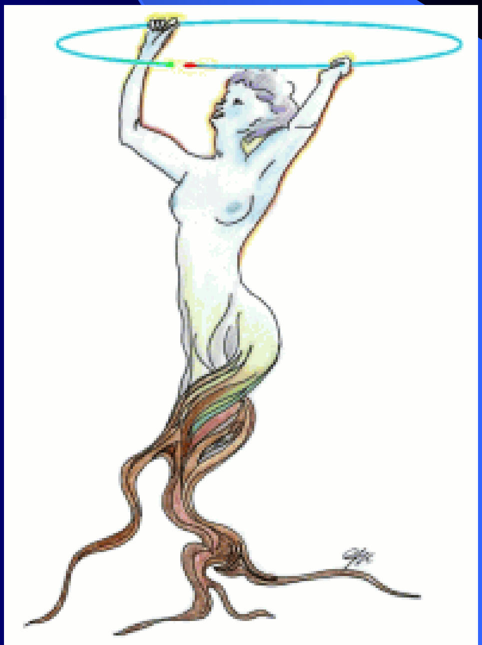
Guidelines for the software framework

- Easy interface with existing packages:
  - Geant3 , Geant4, Fluka, Event generators, HPSS etc
- Simple structure to be used by non-computing experts
- Portability
- Scalability
- Experiment-wide framework
- Use a world-wide accepted framework, if possible

Collaboration-specific framework is less likely to survive in the long term

Proposal for the Infrastructure: ROOT

- Fully OO framework, including:
  - all needed functionalities present (from data taking to final plots)
  - HEP classes (for modular, OO programming) ideal for decentralization of code developers
  - integrated files and objects I/O
  - transparent LAN and WAN support (rootd)
  - transparent HPSS support
  - parallel computation (PROOF)
  - compression
  - interface to SQL/RDBMS
  - interface to GEANT3
  - documentation tools



One Single and Modular framework

- persistency, containers
- histogramming services
- UI, GUI, 2-d, 3-d graphics
- C++ interpreter and scripting language + dynamic compilation and linking
- same for interactive and batch
- call the interpreter from compiled code (Interactive algorithm debugging)
- coding rules
- reconstruction & analysis are naturally developing in the same framework

...and more

- Extensive CERN support
  – Bonus for small collaborations
- Unprecedented Large contributing HEP Community
  – Open Source project
- Multiplatforms
- Support multi-threading and asynchronous I/O
  – Vital for a reconstruction farm
- Optimised for different access granularity
  – Raw data, DST's, NTuple analysis

LAN/WAN files

- *Files and Directories*
  - *a directory holds a list of named objects*
  - *a file may have a hierarchy of directories (a la Unix)*
  - *ROOT files are machine independent*
  - *built-in compression*
- *Support for local, LAN and WAN files*
  - *TFile f1("myfile.root")*
  - *TFile f2("http://pcbrun.cern.ch/Renefile.root")*
  - *TFile f3("root://cdfsga.fnal.gov/bigfile.root")*
  - *TFile f4("rfio://alice/run678.root")*

Local file

Remote file access via a Web server

Remote file access via the ROOT daemon

Access to a file on a mass store hpps, castor, via RFIO

Support for HSM Systems

- Two popular HSM systems are supported:
  - CASTOR
    - developed by CERN, file access via RFIO API and remote rfiod
  - dCache
    - developed by DESY, files access via dCache API and remote dcached

```
TFile *rf = TFile::Open("rfio://castor.cern.ch/alice/aap.root")
TFile *df = TFile::Open("dcache://main.desy.de/h1/run2001.root")
```

PROOF

- Data Access Strategies
  - Each slave get assigned, as much as possible, packets representing data in local files
  - If no (more) local data, get remote data via rootd and rfio (needs good LAN, like GB eth)

- The PROOF system allows:
  – parallel analysis of trees in a set of files
  – parallel analysis of objects in a set of files
  – parallel execution of scripts

<u>on clusters of heterogeneous machines</u>

3-D Graphics

- Basic primitives
  - TPolyLine3D, TPolyMarker3D, THelix, TMarker3DBox,TAxis3D
- Geant primitives
  - Support for all Geant3 volumes + a few new volume types
  - TBRIK,TCONE,TCONS,TCTUB,TELTU,TGTRA,THYPE, TPARA,TPCON, TPGON,TSPHE,TTUBE,TTUBS,TTRAP,TTRD1,TTRD2,TXTRU
- Rendering with:
  - TPad
  - X3D (very fast. Unix only. Good on networks)
  - OpenGL
  - OpenInventor (new addition in 3.01)

General Architecture: Guidelines

- Ensure high level of modularity (for easy of maintenance)
- Absence of code dependencies between different detector modules (to C++ header problems)
- Design the structure of every detector package so that static parameters (i.e. geometry and detector response parameters) are stored in distinct objects
- The data structure to be built up as ROOT TTree-objects
- Access either the full set of correlated data (i.e., the event) or only one or more sub-sample (one or more detectors).

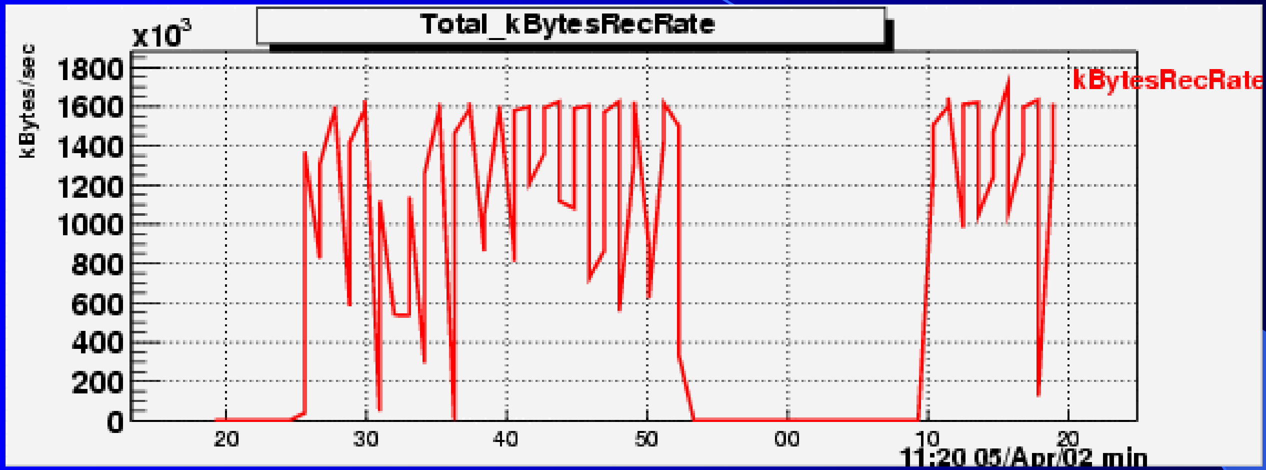
4th Concept simulation & reconstruction Software: IVCroot

- Derived from Aliroot
- Robustness and efficiency now proven
- Architecture and Data Model unmodified
- Some adaptation to the 4th Concept (minor changes)
- Build new modules for 4th Concept specific detectors

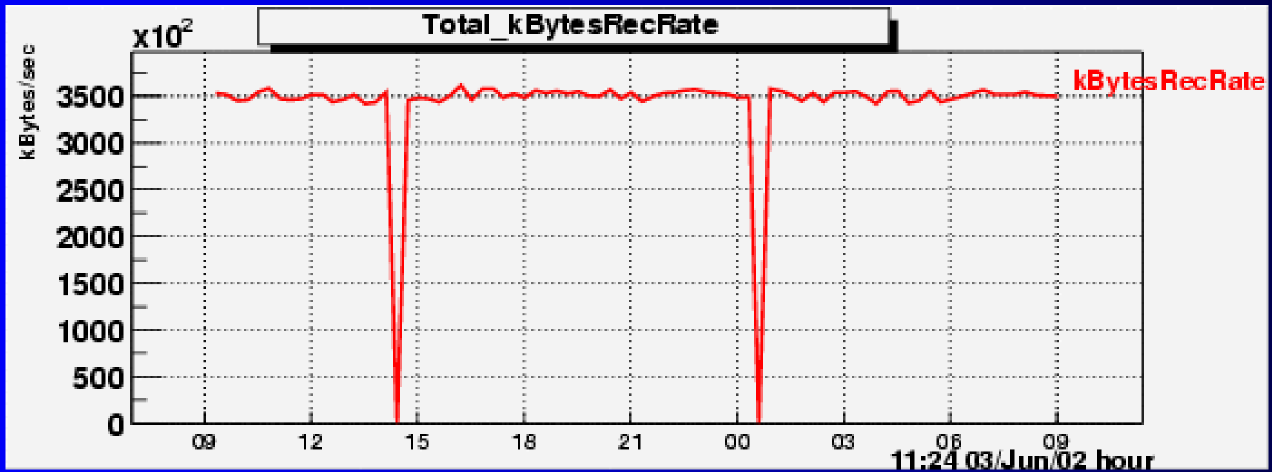
- Leave the Software Engineering to the experts
- Concentrate on the Physics

Alice Performances (MDC IV)

Data generation in LDC, event building, no data recording



Data generation in LDC, event building, data recording to disk



Total: 192 CPU servers (96 on Gbe, 96 on Fe), 36 DISK servers, 10 TAPE servers

Building a Modular System

Use ROOT's Folders

Folders Types

- Data
  - Constants
  - Event
- Tasks
  - Tasks can be organized into a hierarchical tree of tasks and displayed in the browser.
  - A Task is an abstraction with standard functions to Begin,Execute,Finish.
  - Each Task derived class may contain other Tasks that can be executed recursively

Folders Interoperate

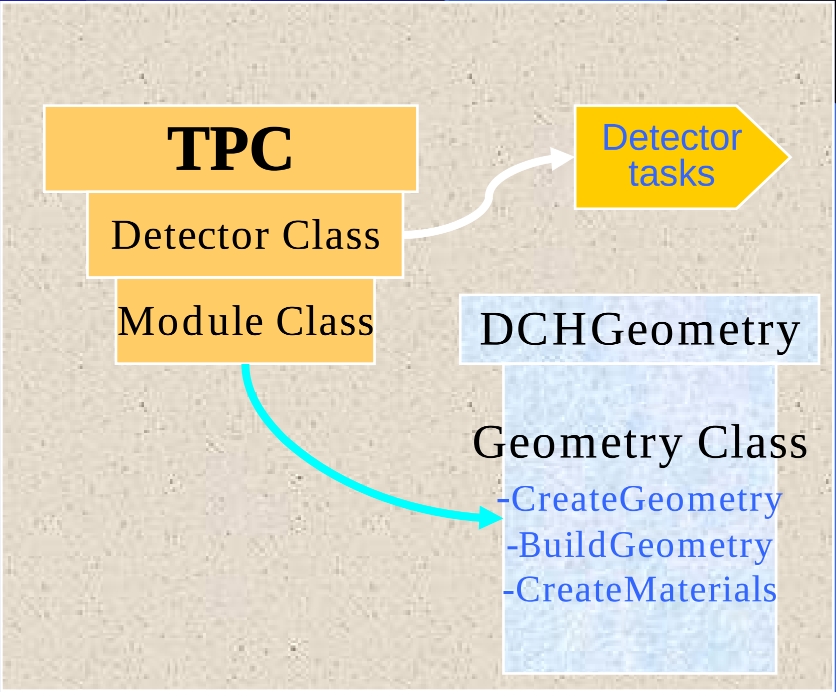
- Data Folders are filled by Tasks (producers)
- Data Folders are used by Tasks (consumers)

Coordinating Tasks & Data

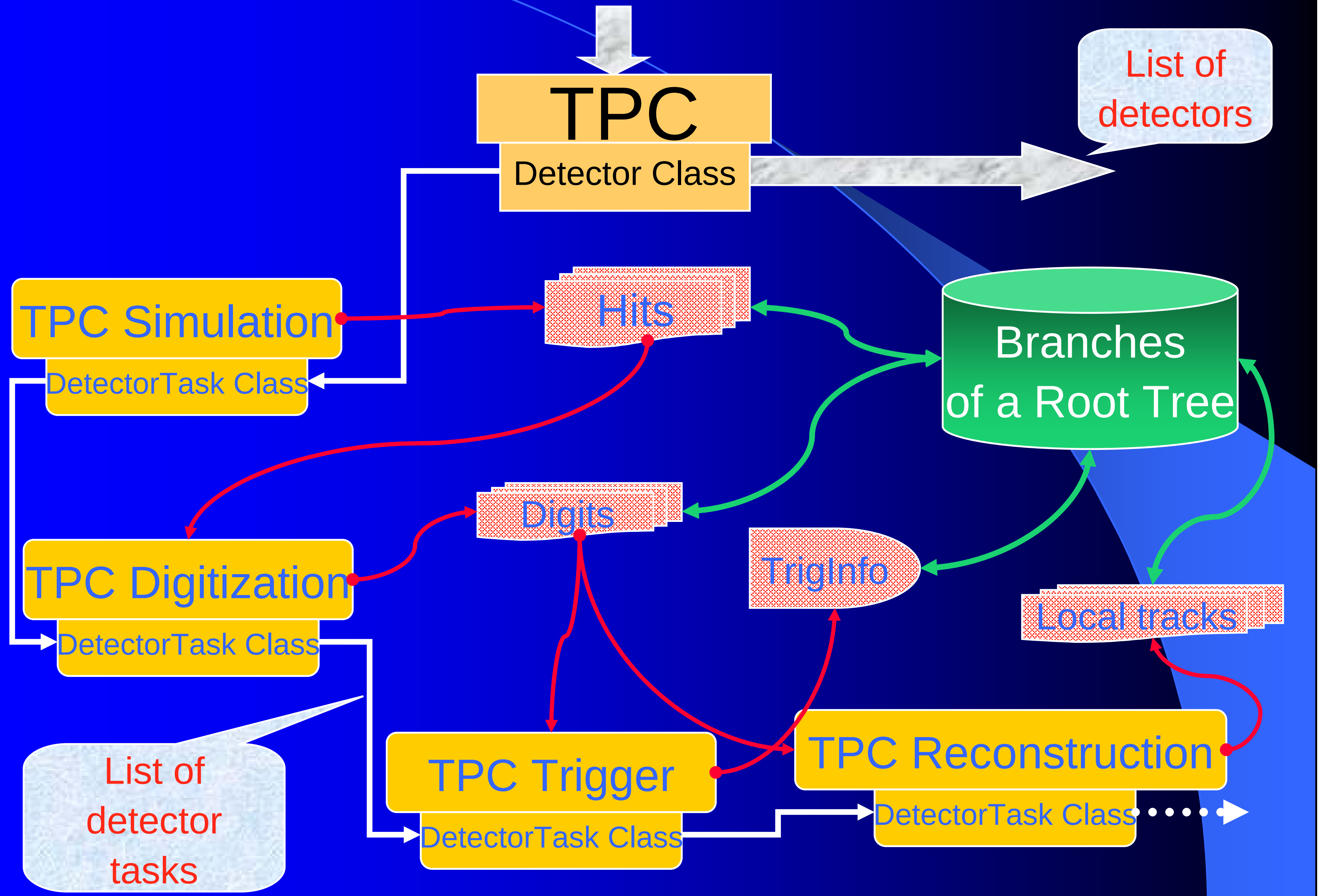
- Detector stand alone (Detector Objects)
  - Each detector executes a list of detector actions/tasks
  - On demand actions are possible but not the default
  - Detector level trigger, simulation and reconstruction are implemented as clients of the detector classes
- Detectors collaborate (Global Objects)
  - One or more Global objects execute a list of actions involving objects from several detectors
- The Run Manager
  - executes the detector objects in the order of the list
  - Global trigger, simulation and reconstruction are special services controlled by the Run Manager class
- The Offline configuration is built at run time by executing a ROOT macro

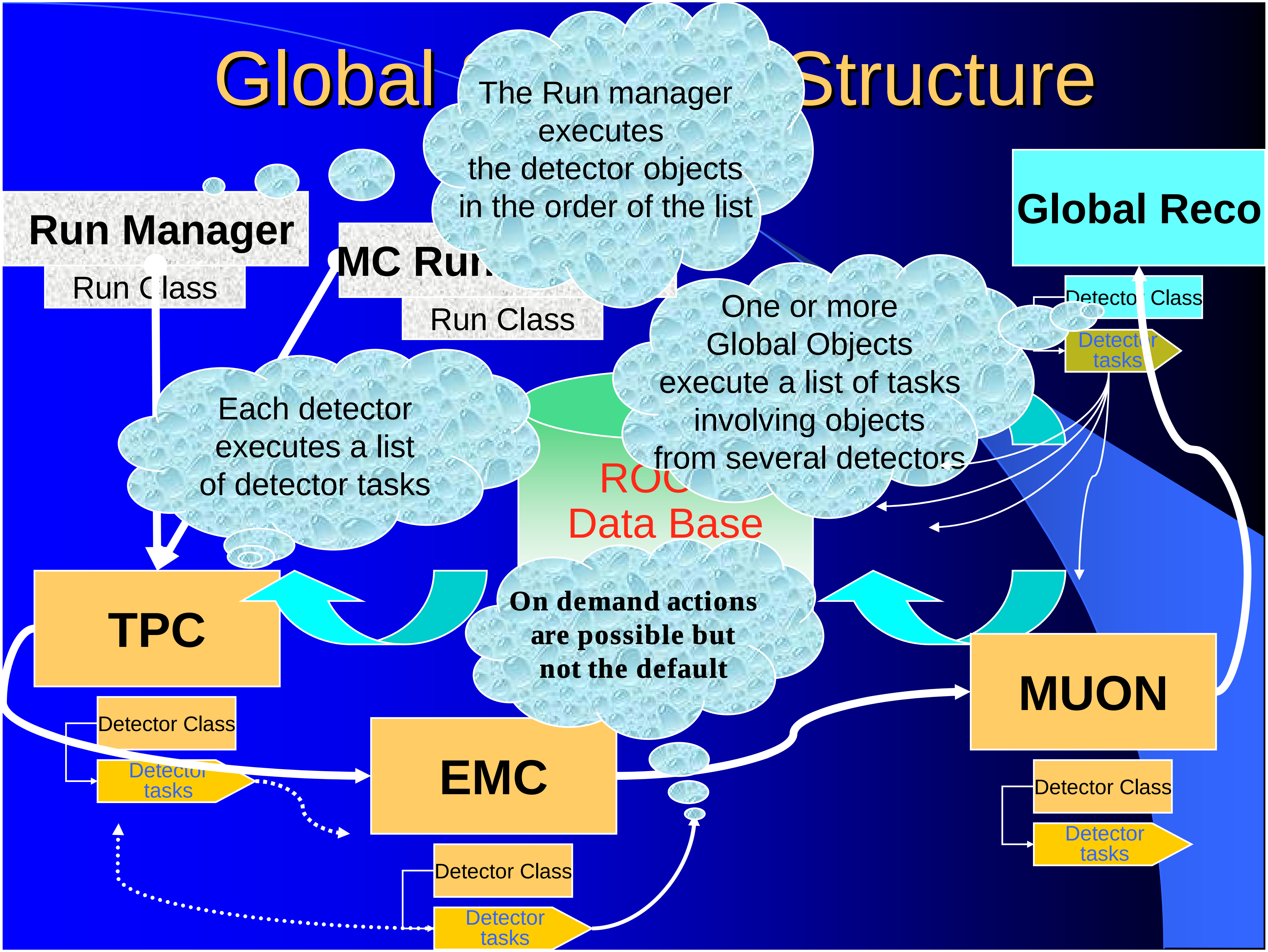
The Detector Class

- **Base class for subdetectors modules.**
- **Both sensitive modules (detectors) and non-sensitive ones are described by this base class. This class**
- **supports the hit and digit trees produced by the simulation**
- **supports the the objects produced by the reconstruction.**
- **This class is also responsible for building the geometry of the detectors and the event display.**
- **Several versions of the same detector are possible (choose at run time)**



Detector Level Structure

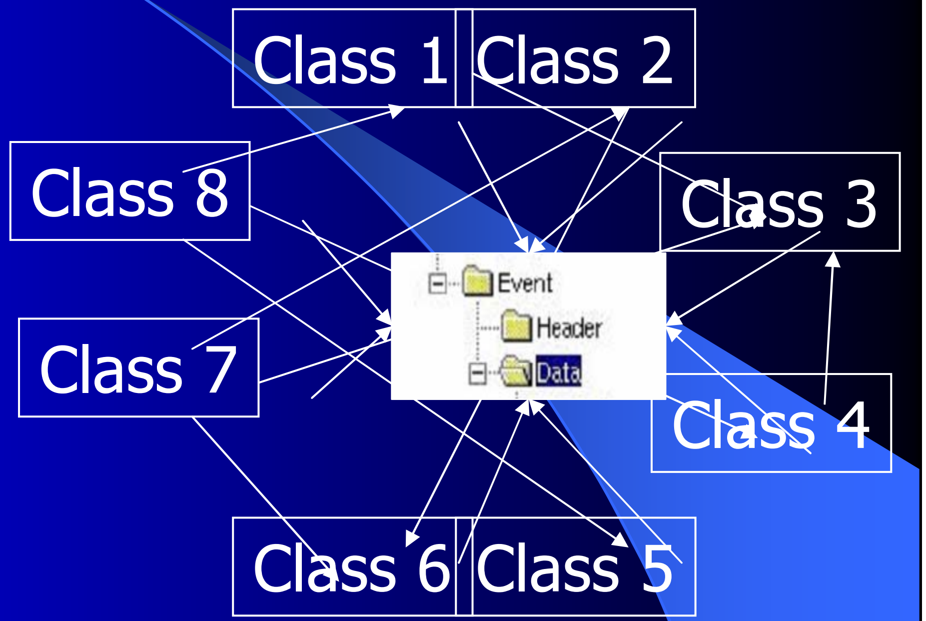




Global
Structure
The Run manager executes the detector objects in the order of the list
Run Manager
Run Class
MC Run
Run Class
Global Reco
Detector Class
Detector tasks
One or more Global Objects execute a list of tasks involving objects from several detectors
Each detector executes a list of detector tasks
Data Base
On demand actions are possible but not the default
TPC
Detector Class
Detector tasks
EMC
Detector Class
Detector tasks
MUON
Detector Class
Detector tasks

Run-time Data-Exchange

- Post transient data to a white board
- Structure the whiteboard according to detector sub-structure & tasks results
- Each detector is responsible for posting its data
- Tasks access data from the white board
- Detectors cooperate through the white board



Montecarlo Organization

The Virtual Montecarlo

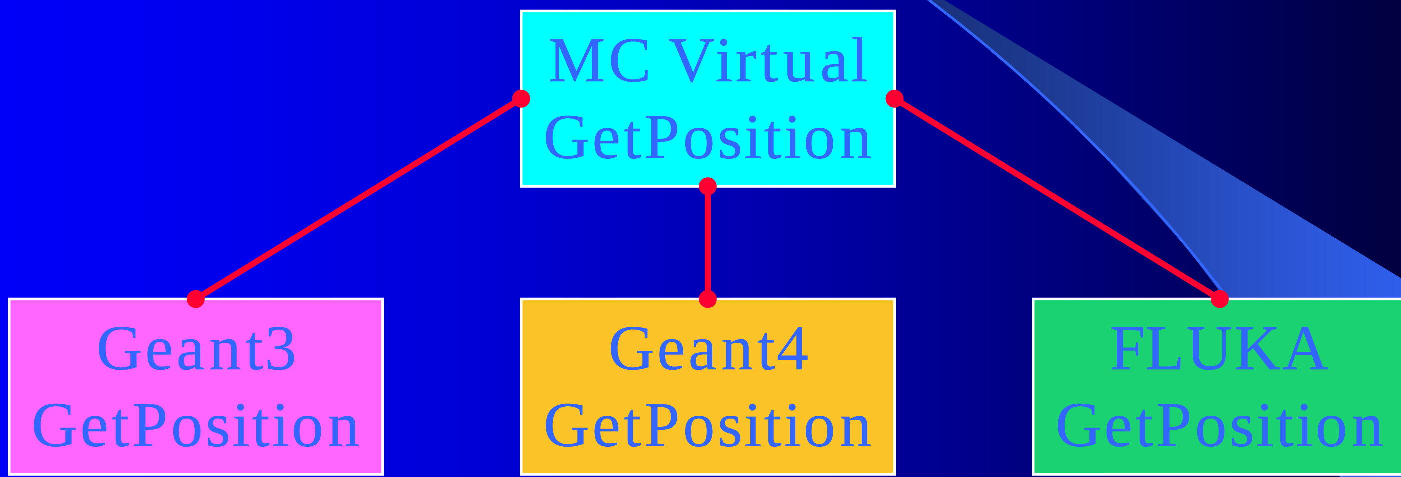
Geant3/Geant4/Fluka Interface

Generator Interface

The Virtual MC Concept

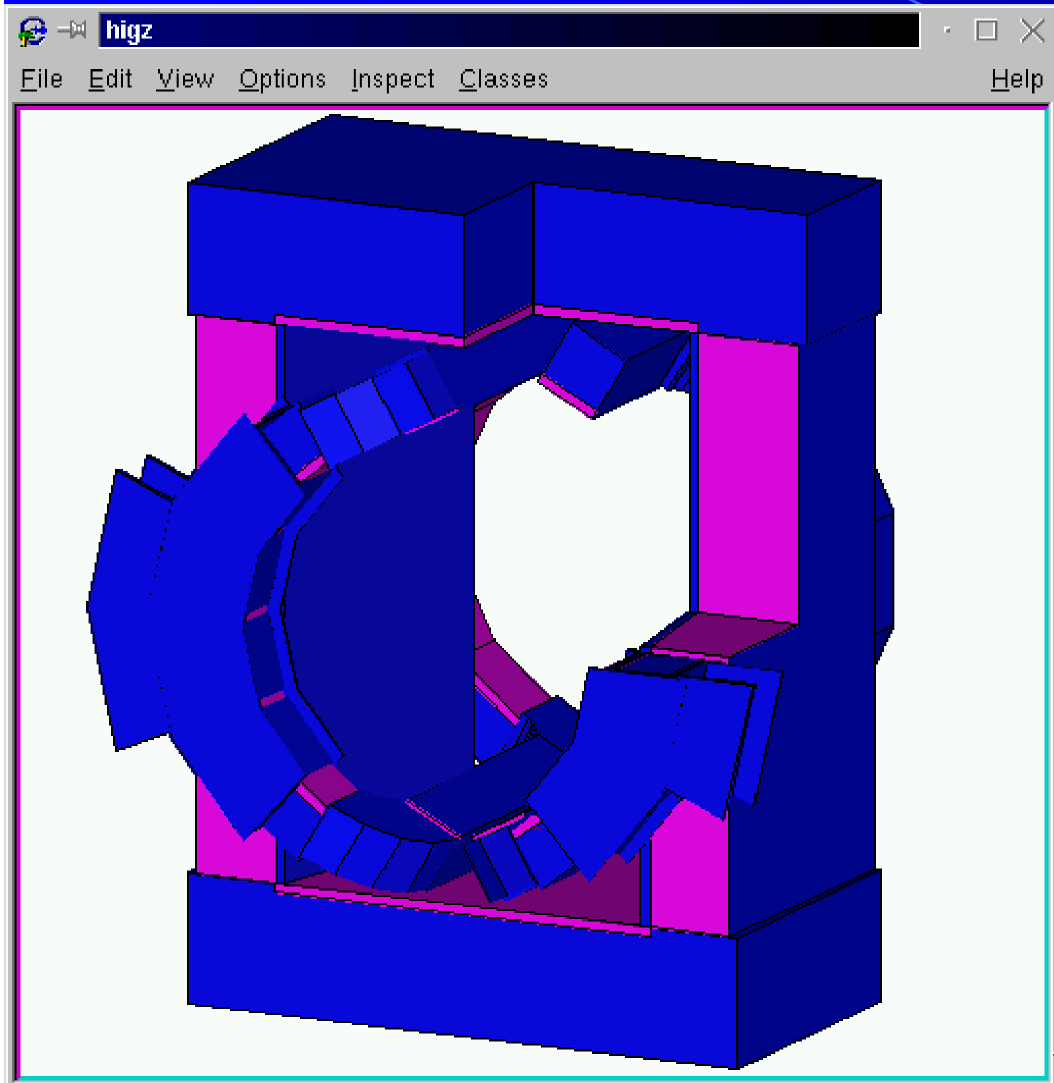
- Virtual MC provides a virtual interface to Monte Carlo
- It decouples the dependence of a user code on a concrete MC
- It allows to run the same user application with all supported Monte Carlo programs
- The concrete Monte Carlo (Geant3, Geant4, Fluka) is selected and loaded at run time
- It allows the comparison between Geant3 and Geant4 using the same geometry and data structure (QA)
- Smooth transition to Geant4 with maximum reuse of Geant3 based simulation (user-) code
- Ideal when switching from a fast to a full simulation: VMC allows to run different simulation Monte Carlo from the same user code

Example: Calling a Virtual Function

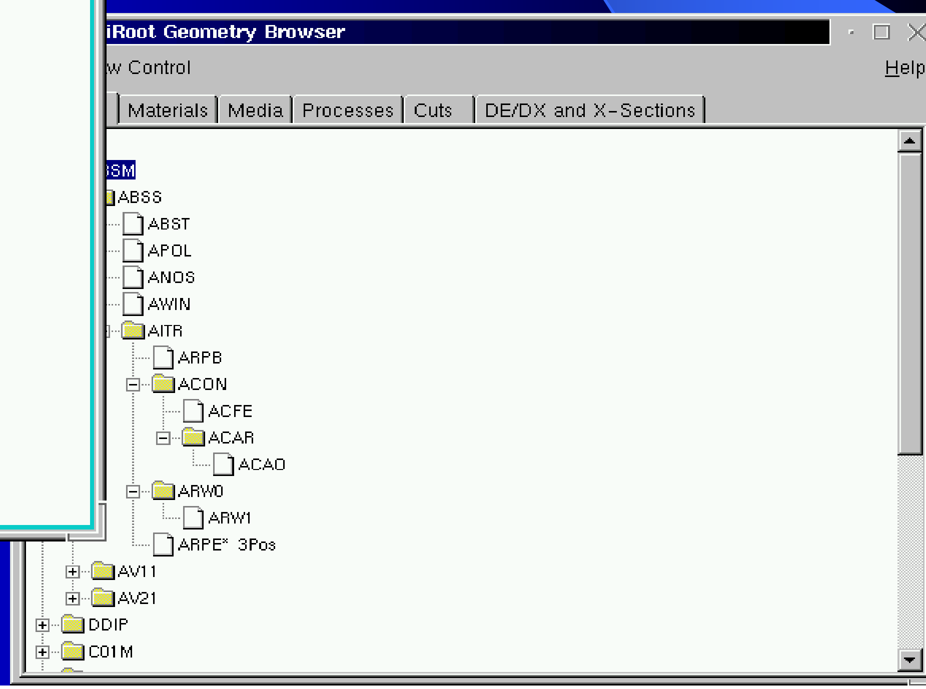


MC->GetPosition(Float_t *x)

Detector Geometry in VMC



- The geometry can be specified using:
  - Root (TGeo)
  - Geant3
  - Geant4
  - Fluka
  - XML
  - Oracle
  - CAD



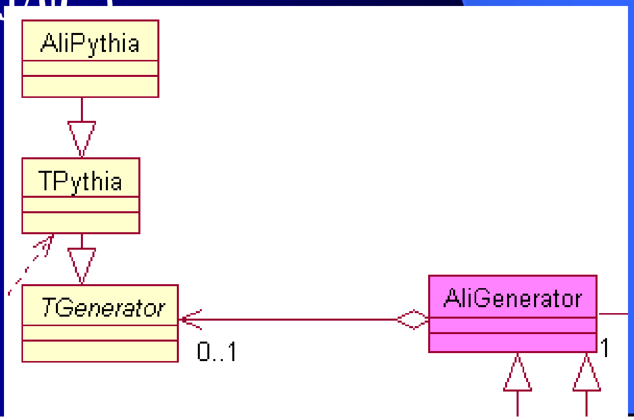
Generator Interface

- ***TGenerator*** is an abstract base class, that defines the interface of ROOT and the various event generators (thanks to inheritance)
- Provide user with
  - Easy and coherent way to study variety of physics signals
  - Testing tools
  - Background studies
- Possibility to study
  - Full events (event by event)
  - Single processes
  - Mixture of both (“Cocktail events”)
- Easily interface with existing fortran generators
- Many already existing: Pythia, Jetset, Herwig, etc.
- Can also import/export plain text files

An Example: Pythia and Jetset

- TPythia derived from TGenerator
  - Access to Pythia and Jetset common blocks via class methods
  - implements TGenerator methods

- MyPythia derived from TPythia
  - High level interface to Jetset and Pythia
  - Tailored to our special needs:
    - generation of hard processes (charm, beauty, J/ψ …)
    - selection of structure function
    - forced decay modes
    - particle decays … and more



One Step Up: GenCocktail

- Generation of Cocktail of different processes
  - Generation from parameterised transverse momentum and rapidity
  - Decays using JETSET
  - Rate and weighting control
  - Allow easy mixing of signal and background
- Mix digits generated with different Montecarlo's
  - Ex.: Signal with G4 + background with Fluka

Summary of Features

- IVCRoot (from after AliRoot) is an highly modular system
- The basic framework (developed by R. Brun an F. Carminati) is robust and efficient
- It takes care of the I/O system (persistent data) and of the steering of the modules
- Each detector is represented by an independent module developed by the user on the base od a virtual detector class
- Persistent data structure can be modified to meet detector specific needs
- Multiple versions (same or different detector: ex. TPC vs DCH) can live togheter
- Actual version is choosen at run time (no need to recompile)
- Perfect for an ILC study environnment

- **Cons: long learning curve (about three months)**

Interface to other ILC Software

- Detector Geometry can be exchanged with the system developed at SLAC trough the VGM
- The Virtual Geometry Modeller (VGM) has been developed as a generalization of the existing convertors roottog4, g4toxml provided within Geant4 VMC
- VGM can provide format exchange among several systems (Geant4, Root, XML AGDD, GDML)
- The implementation of the VGM for a concrete geometry model represents a layer between the VGM and the particular native geometry.
- At present this implementation is provided for the Geant4 and the Root TGeo geometry models.
- Not an issue if the Geometry is described using Root TGeo
- At present, digitization of geometry imported in IVCRoot needs to be coded within the system
- SLAC’s extended GDML (for digitization) could be implemented for automatic digitization

IVCroot status

- IVCroot is already working for the IV Concept design
- All the machinery is in place
- All simulation/reconstruction steps are being implemented:
  - Hits production
  - Summable Digits + Digits
  - Pattern recognition
  - Calibration
  - Reconstruction
  - PID
- Analysis is performed outside IVCroot
- Physics results will be out soon

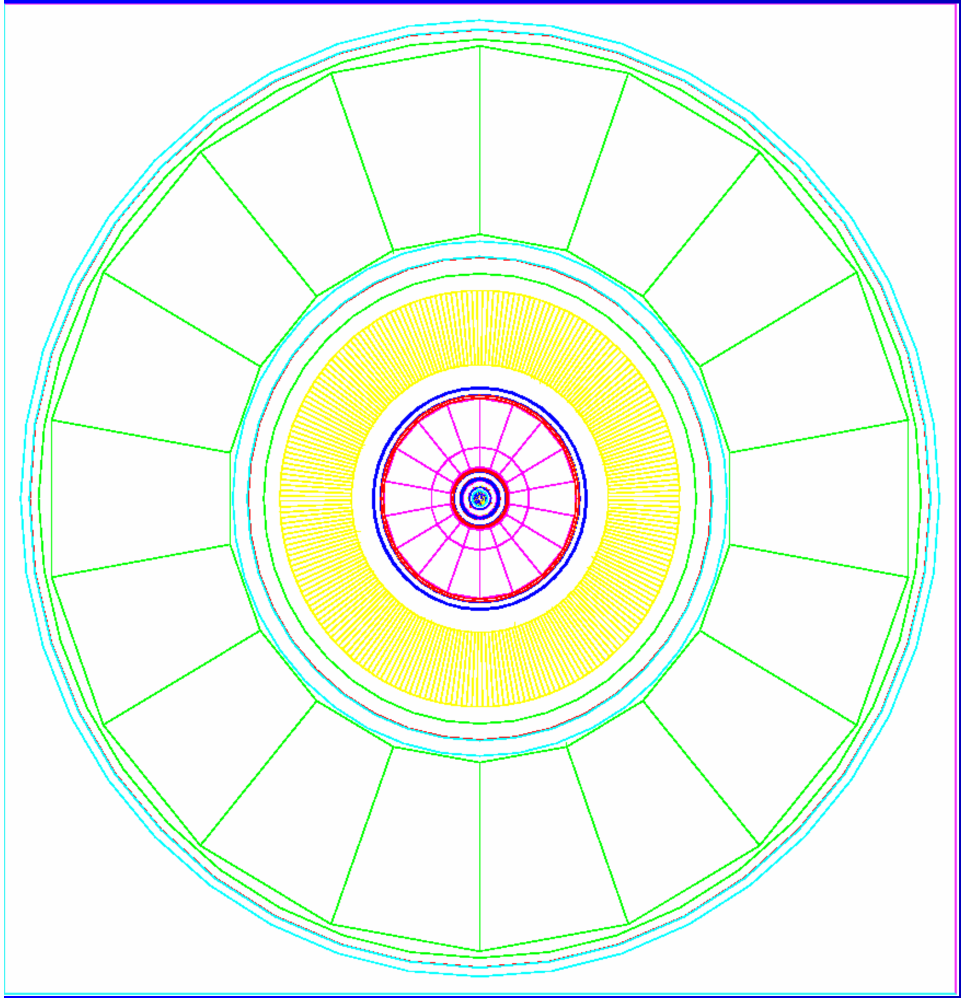
Detector Simulation

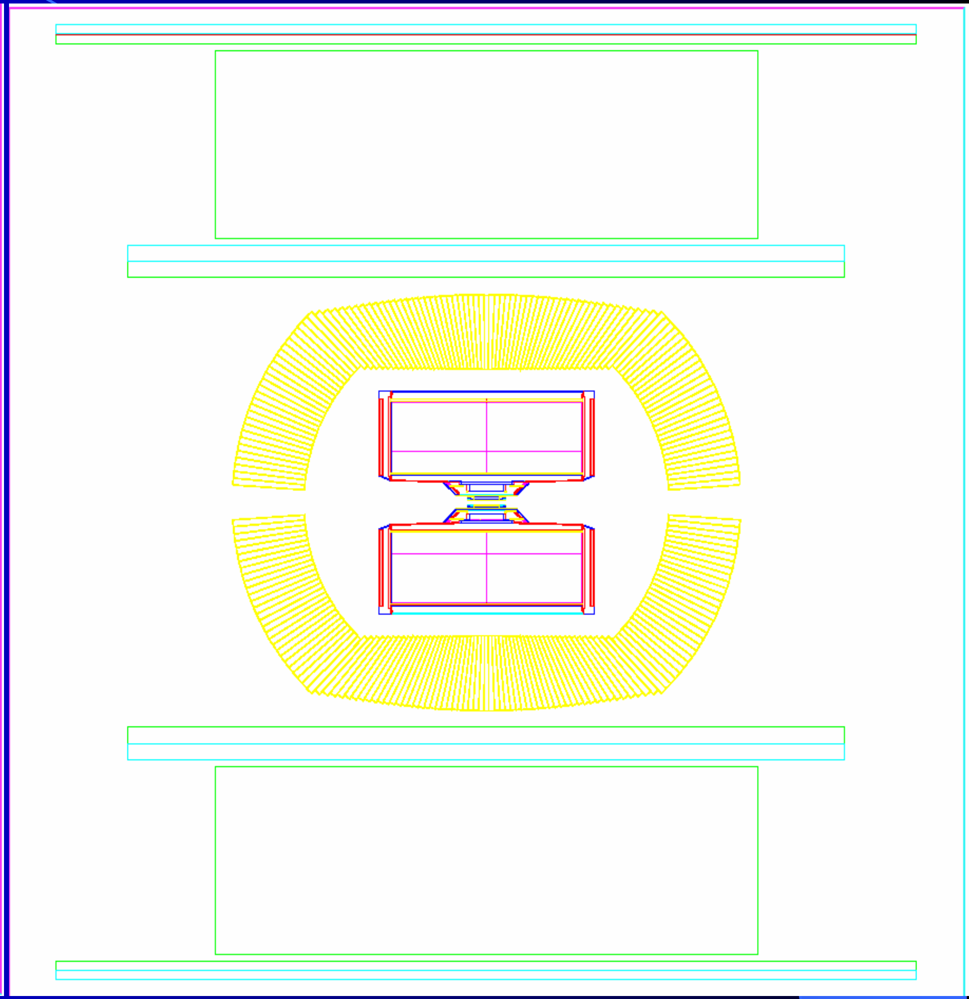
- Vertex Detector
  - r= 3.9, 7.6, 15, 24 cm
  - Resolution: $50/\sqrt{12}$x $50/\sqrt{12}$ μm^2 or $12(r\varphi)x100(z)$ μm^2 + $35(r\varphi)x25(z)$ μm^2
  - Si thickness: 200 μm wafer + 150 μm electronics
  - Material Budget: 0.4% X_o (include termal shield)
- Central Tracker
  - Mock "real" detector (to be replaced by 4th Concept's)
  - Mostly to test framework and Kalman Filter
- ECAL
  - Technology: Lead tungstate crystals (PWO)
  - Crystal size: 2x2x18 cm^3 + 10 cm Electronics/cooling
  - Segmentation: matches the DREAM calorimeter
  - Readout : 5x5 mm^2 APD
  - R_M: 2.2 cm
  - Depth: 20 X_o
  - One Readout crystals (soon to become Dual Readout)

Detector Simulation

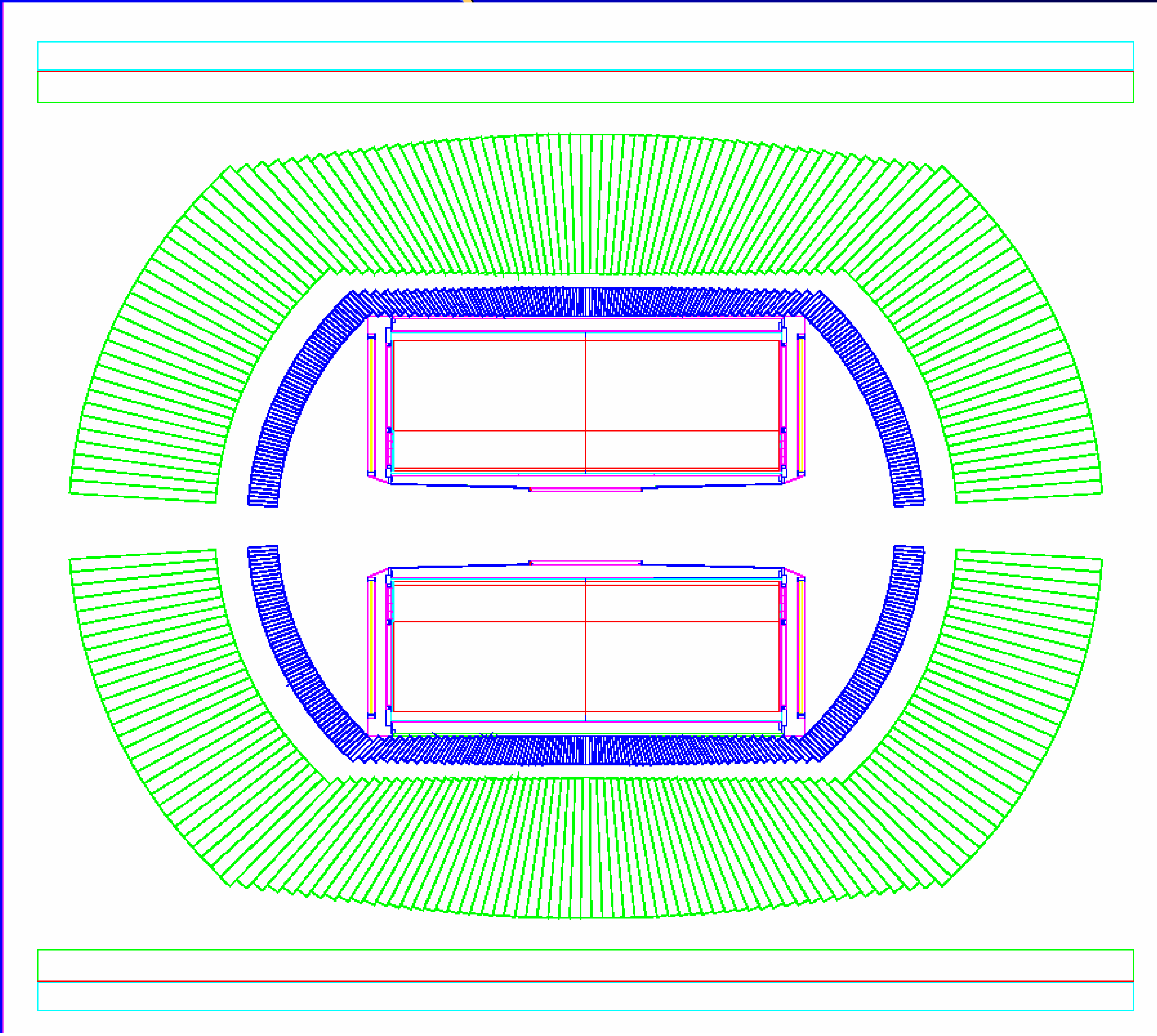
- DREAM
  - No digitization implemented yet
  - Technology: C/S fibers in W or Cu
  - Unit cell: 1x1x100 cm^3 variable across η
  - Unit readout cell: 2x2x100 cm^3
- Muon system
  - "Generic" tracker for the Kalman filter with Pattern Recognition performed by the Central Tracker
  - 18 planes, each 0.7% Xo, 1-15 reading/plane, 200-300 μm
  - Aimed to test tracking in air with -1.5 T field, L*=2.5 m and large $\sqrt{N}$

Geometry Picture

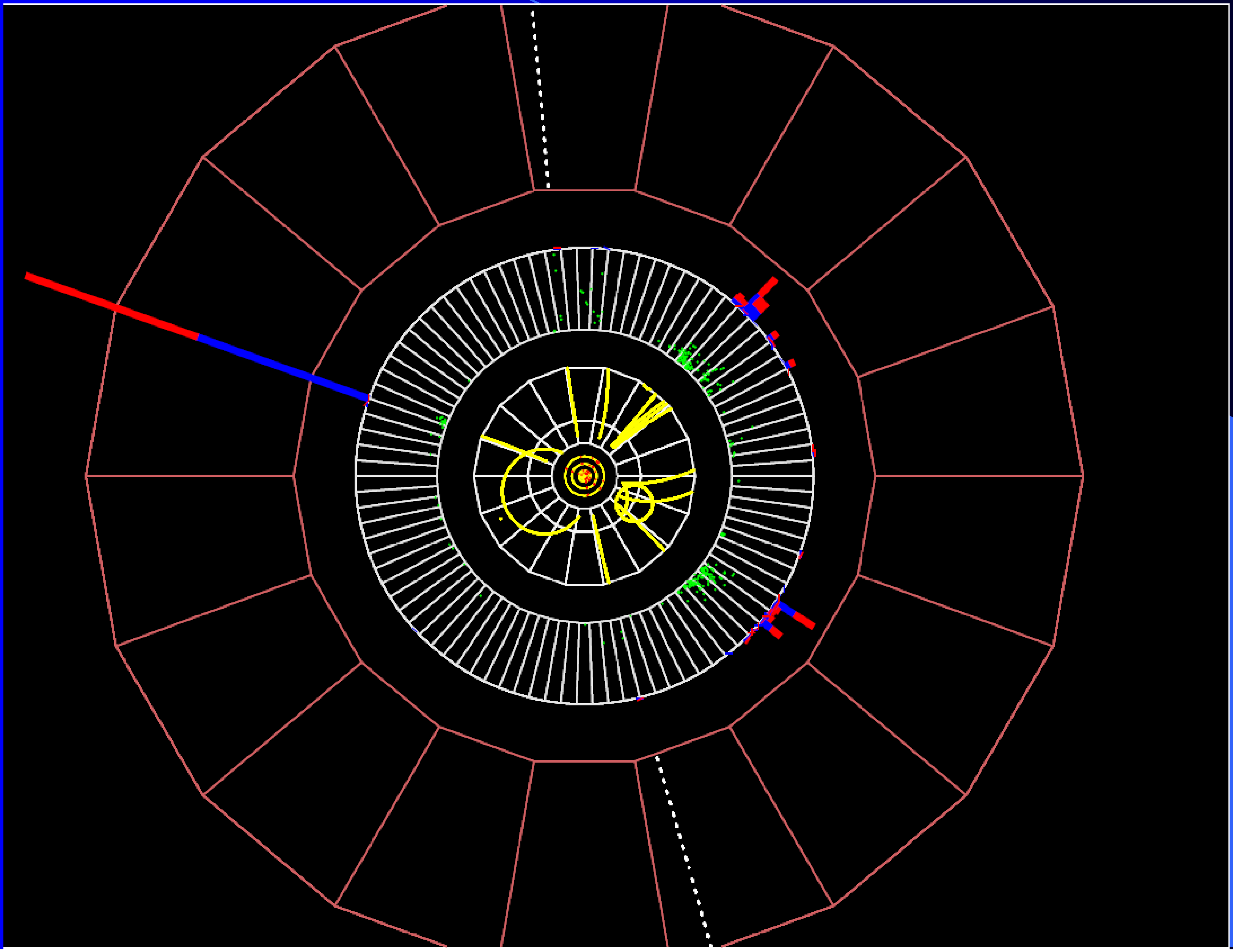




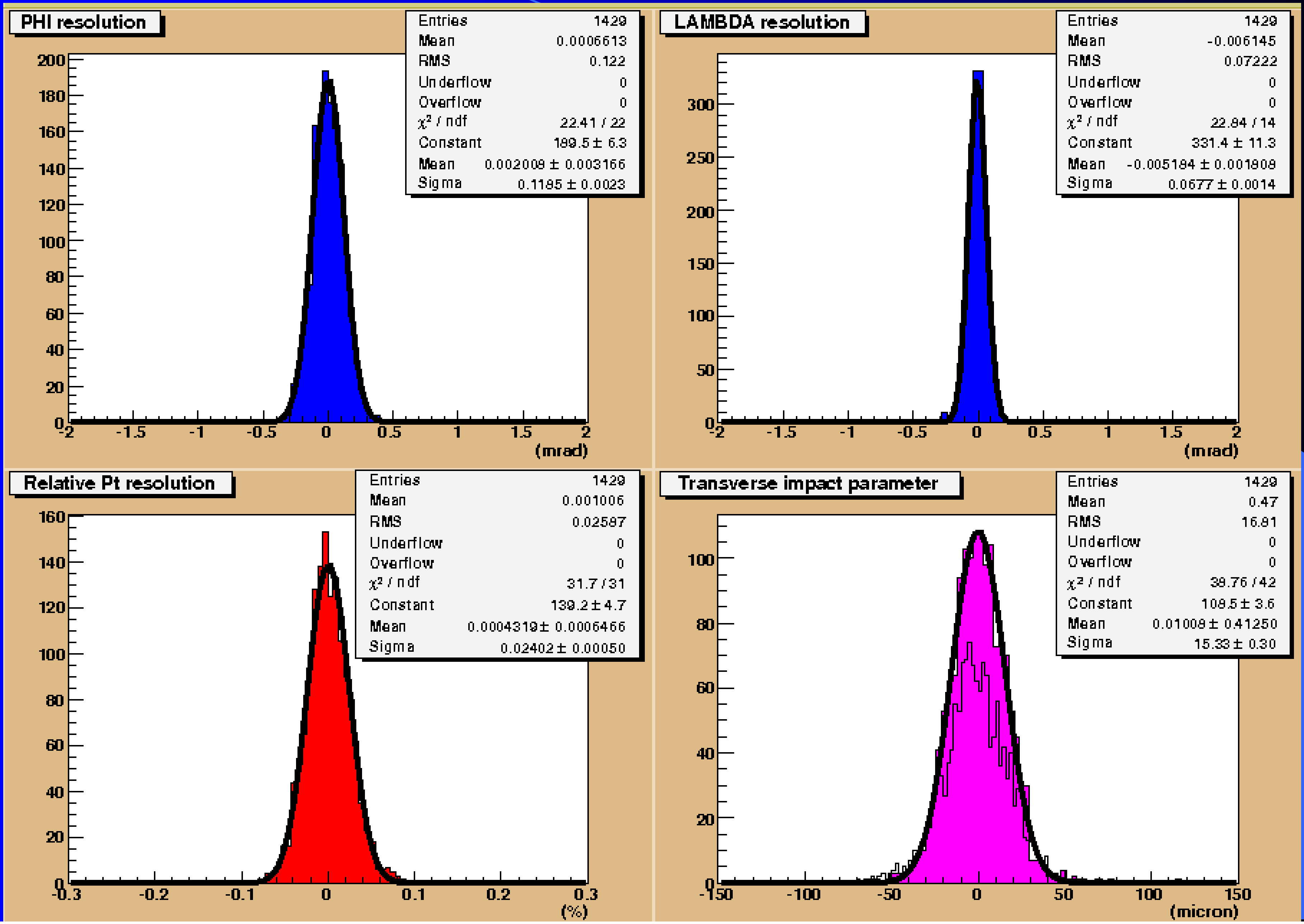
Side View (Inside Inner Coil)



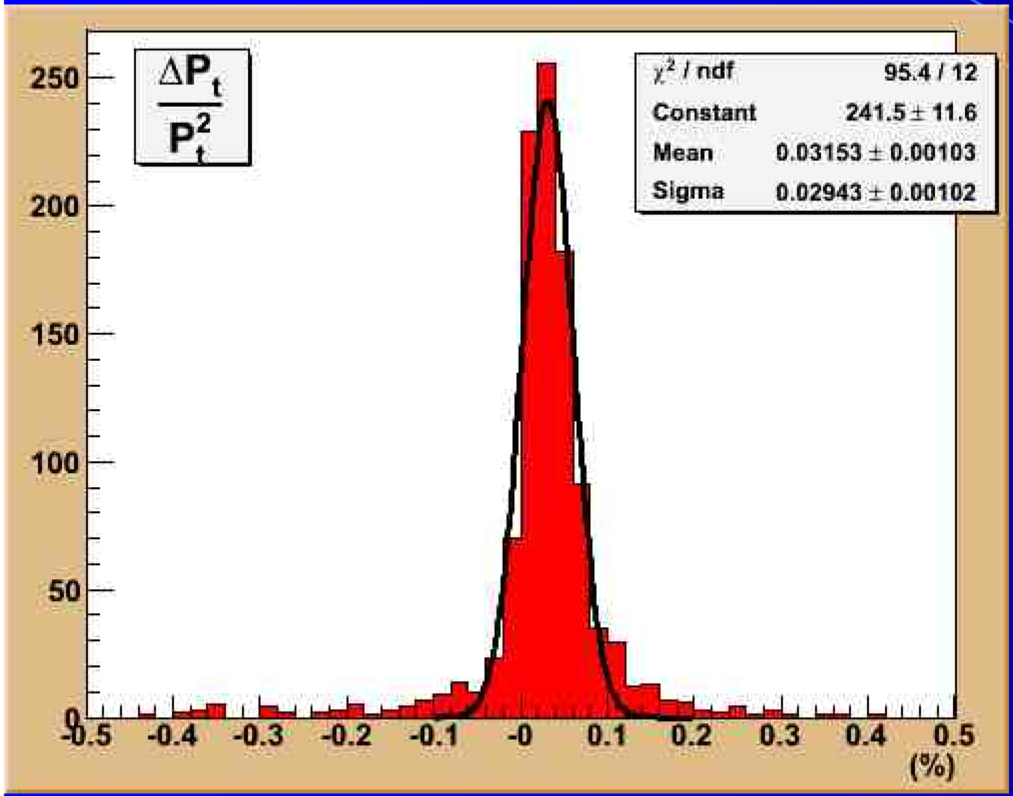
Event display: $e^+e^- \rightarrow H^oZ^o \rightarrow X\mu^+\mu^-$



Pixel Detector + Central Tracker



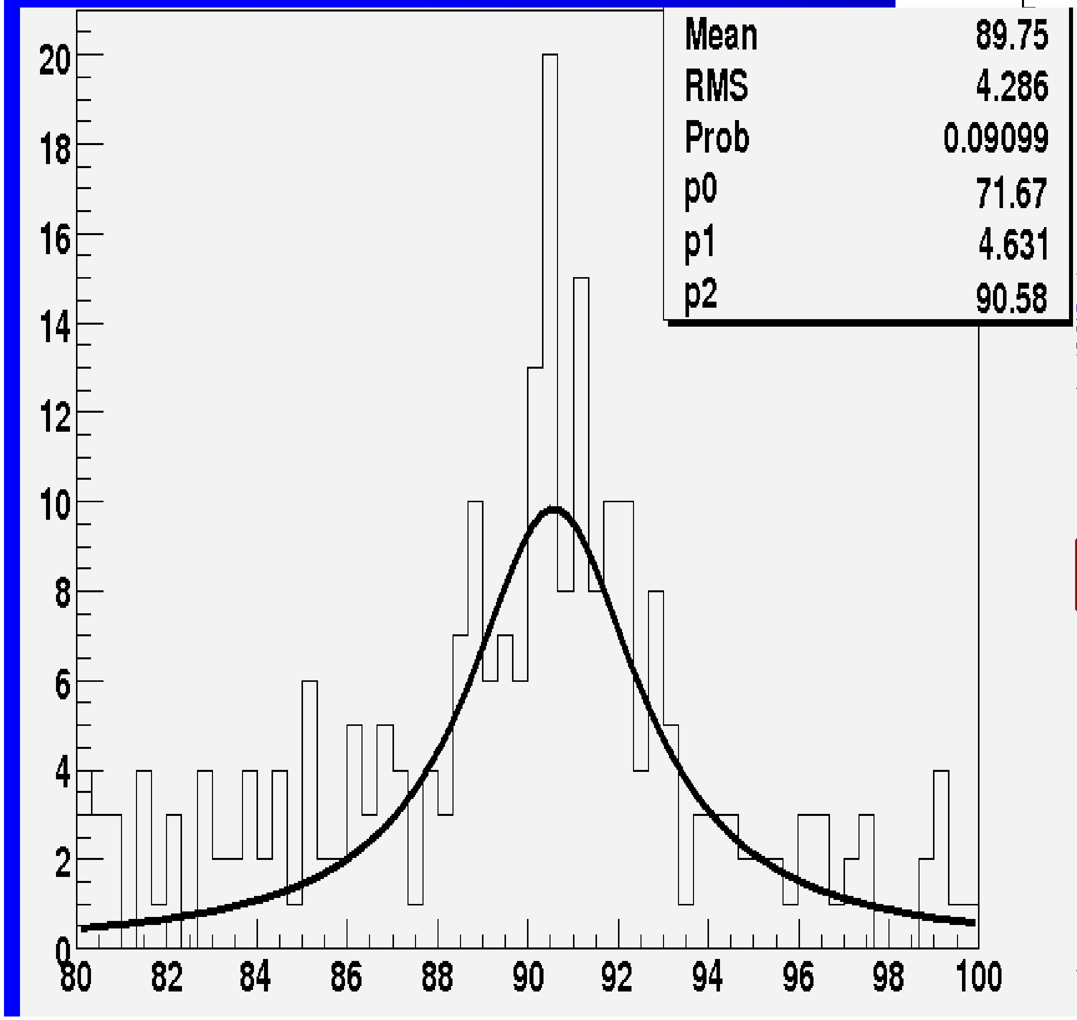
Muon System

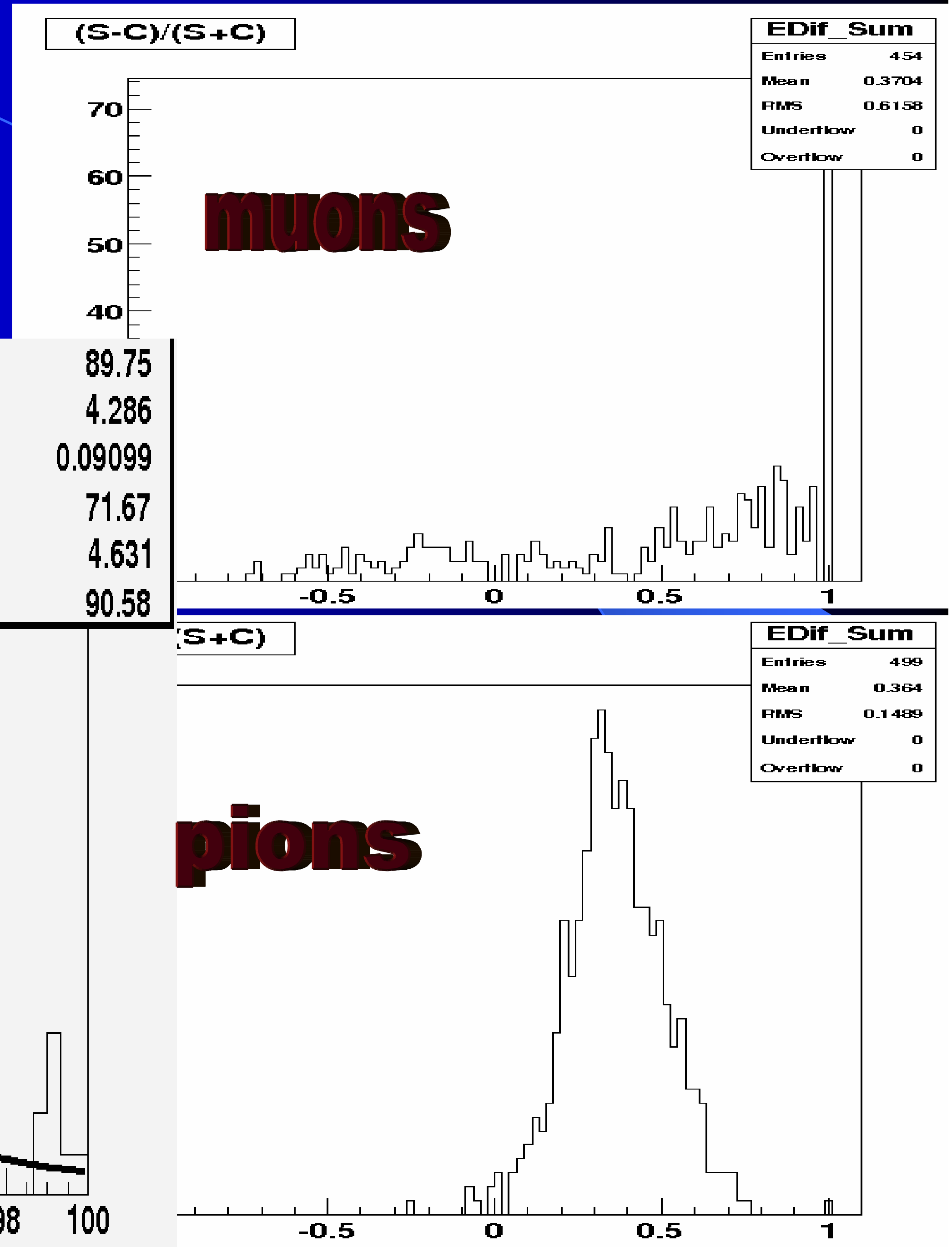


- Central Tracker for pattern recognition + seeds finding
- Kalman filter for trach fit
- Compare momentum reconstructed in the MS to that generated at the origin

Z^o Analysis

- PID from DREAM only





Conclusions

- IVCroot up and running
- Substantial departure from existing ILC code
- But with exchange-program in mind
- Physics results not far away

Backup slides

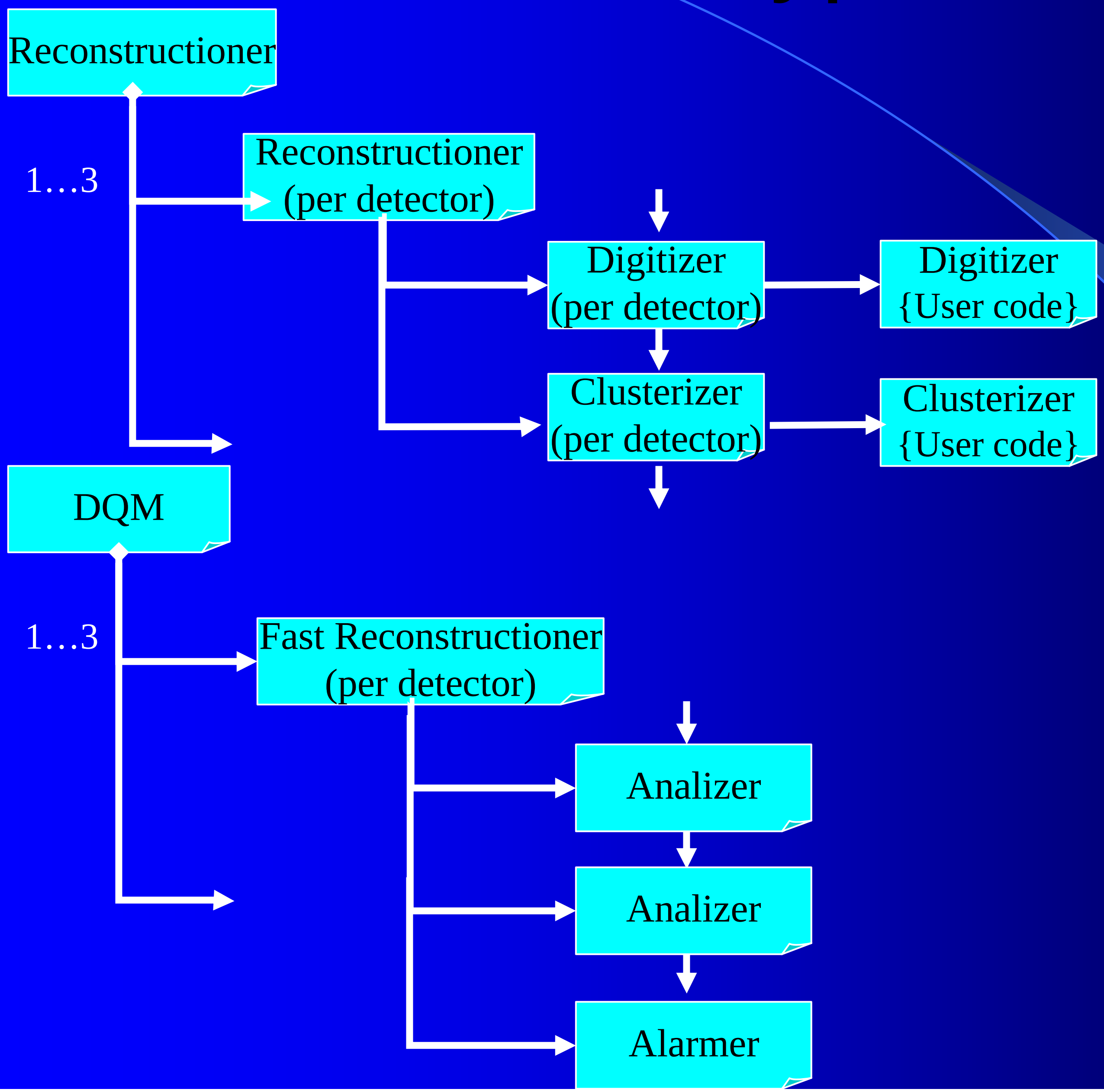
ROOT I/O Performance

- rootd vs nfs I/O test at FNAL: same performane (>50 MB/sec on lan)
- Max throughput over Gigabit Ethernet: 36 MB/s
- Performance improves with chunk size
- Highest throughput achieved: ADC IV (350 MB/sec to 1800 MB/sec)

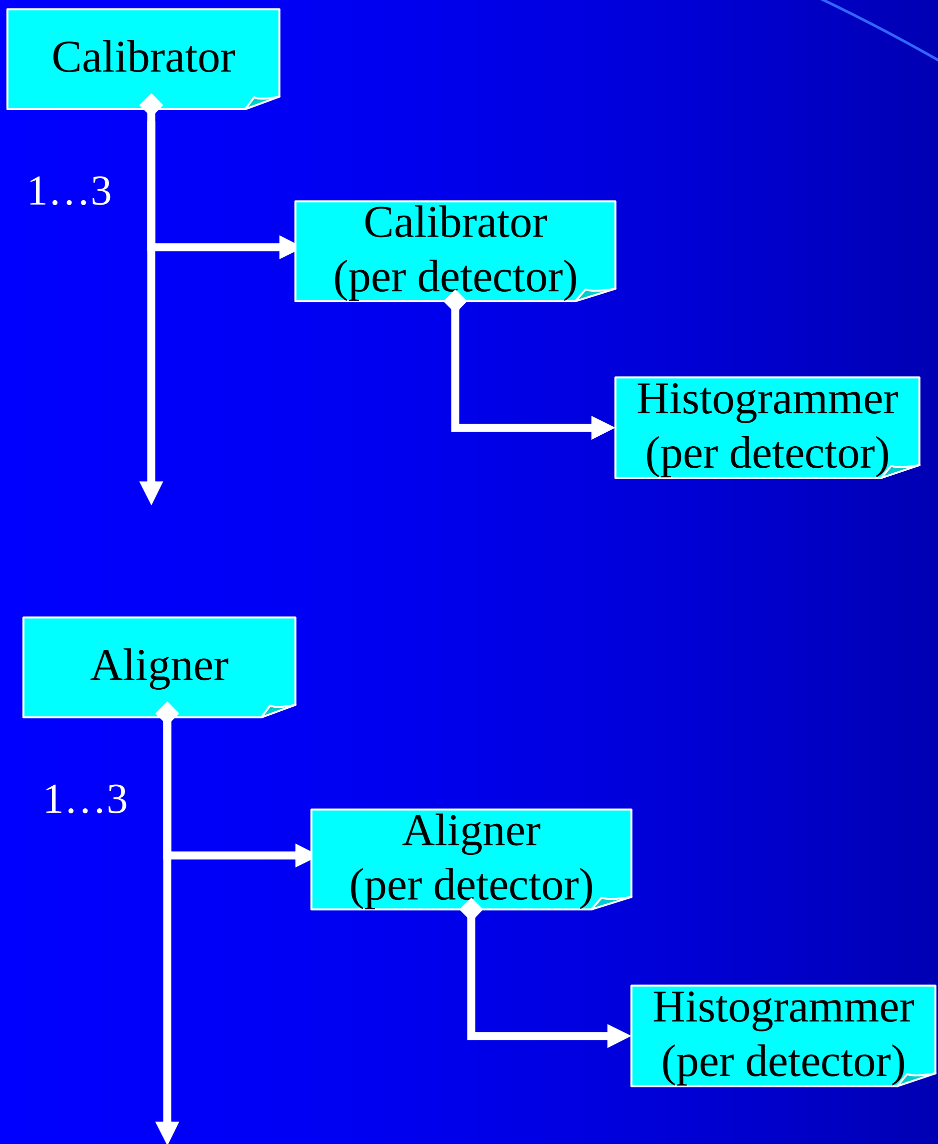
Computing Model: MONARC

- **A central site, *Tier-0***
  - will be hosted by PSI.
- **Regional centers, *Tier-1***
  - will serve a large geographic region or a country.
  - Might provide a mass-storage facility, all the GRID services, and an adequate quantity of personnel to exploit the resources and assist users.
- ***Tier-2* centers**
  - Will serve part of a geographic region, i.e., typically about 50 active users.
  - Are the lowest level to be accessible by the whole Collaboration.
  - These centers will provide important CPU resources but limited personnel.
  - They will be backed by one or several *Tier-1* centers for the mass storage.
  - **In the case of small collaborations, *Tier-1* and *Tier-2* centers could be the same.**
- ***Tier-3 Centers***
  - Correspond to the computing facilities available at different Institutes.
  - Conceived as relatively small structures connected to a reference *Tier-2* center.
- ***Tier-4 centers***
  - Personal desktops are identified as *Tier-4* centers

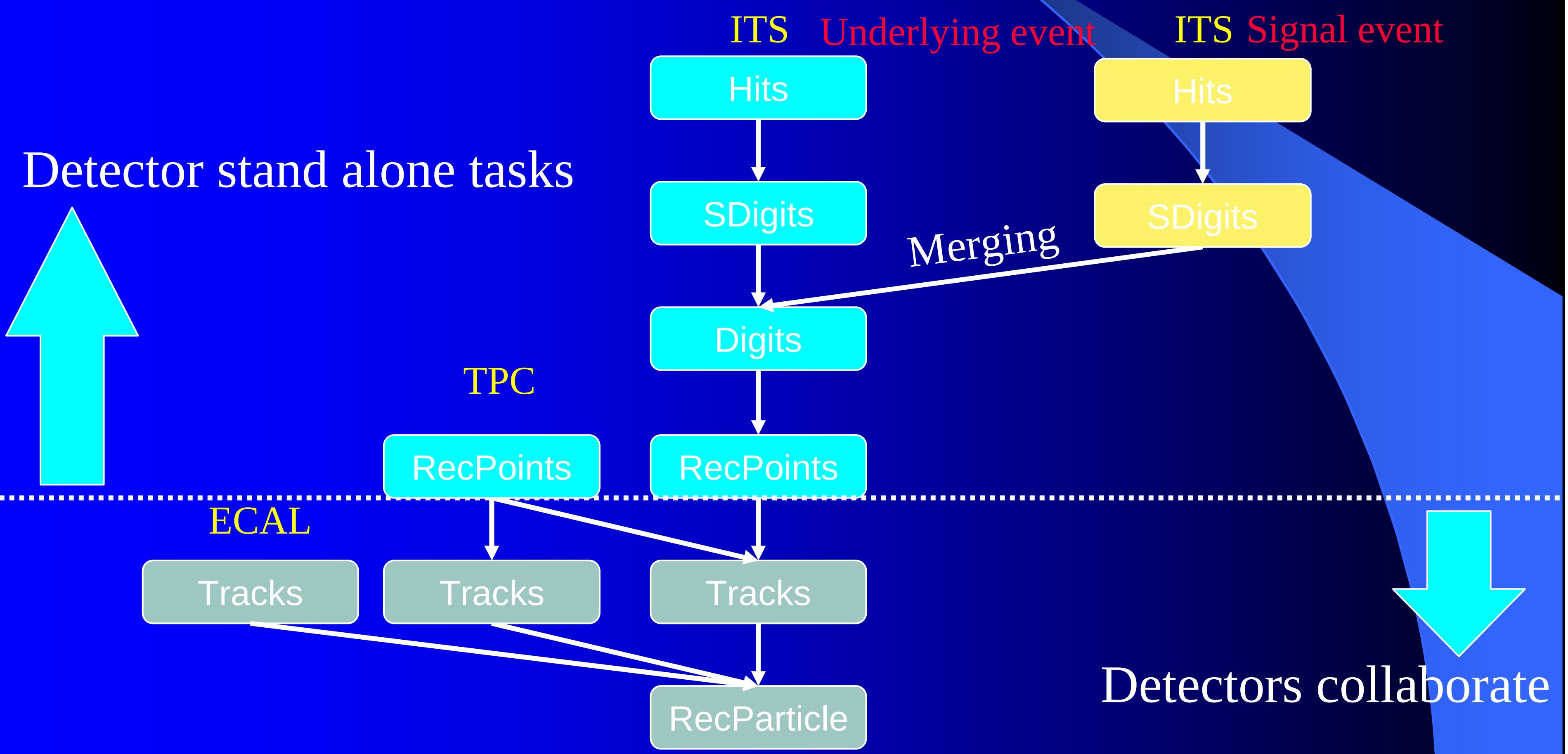
Folders Type: Tasks



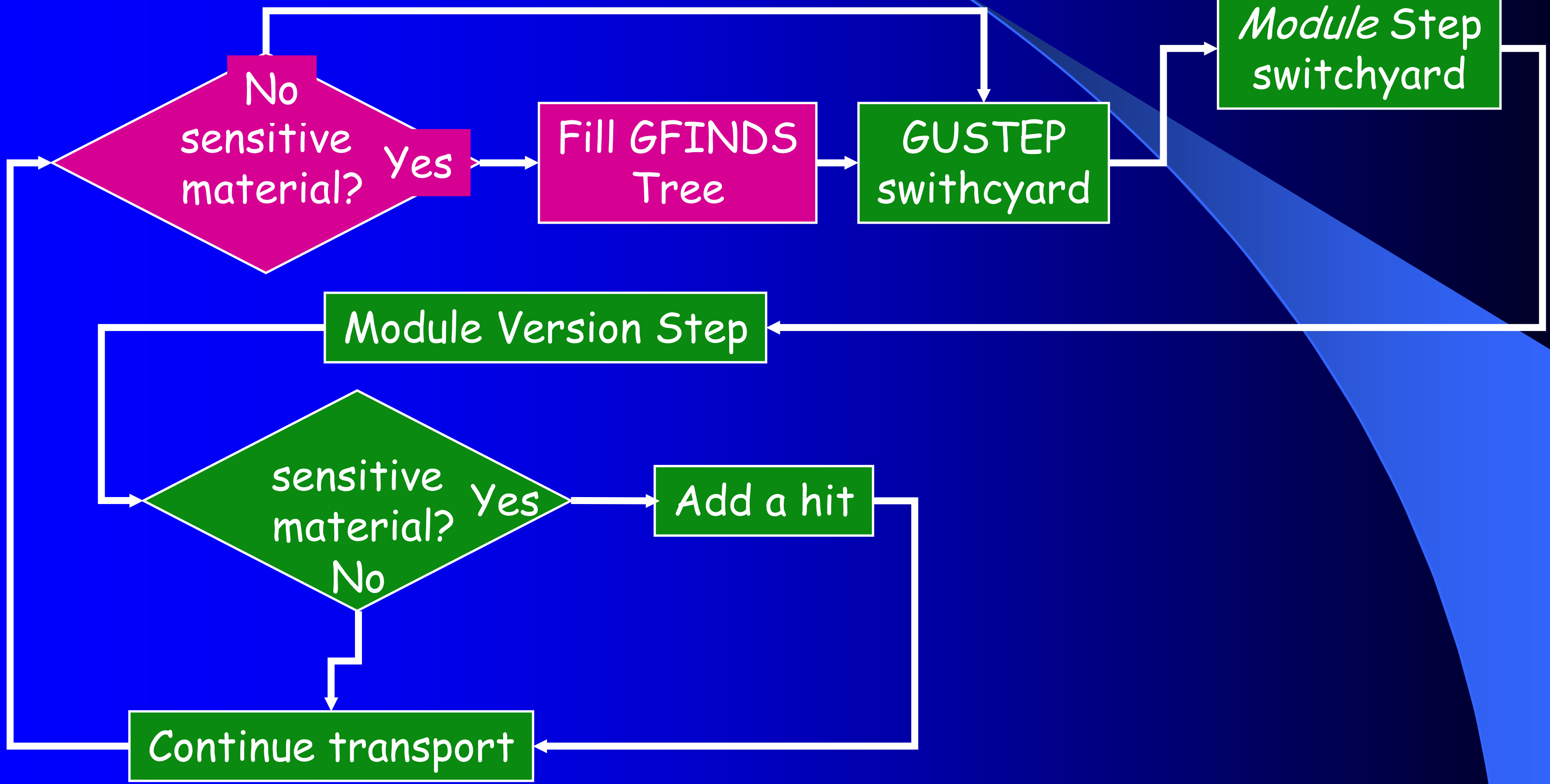
Folders Type: Tasks



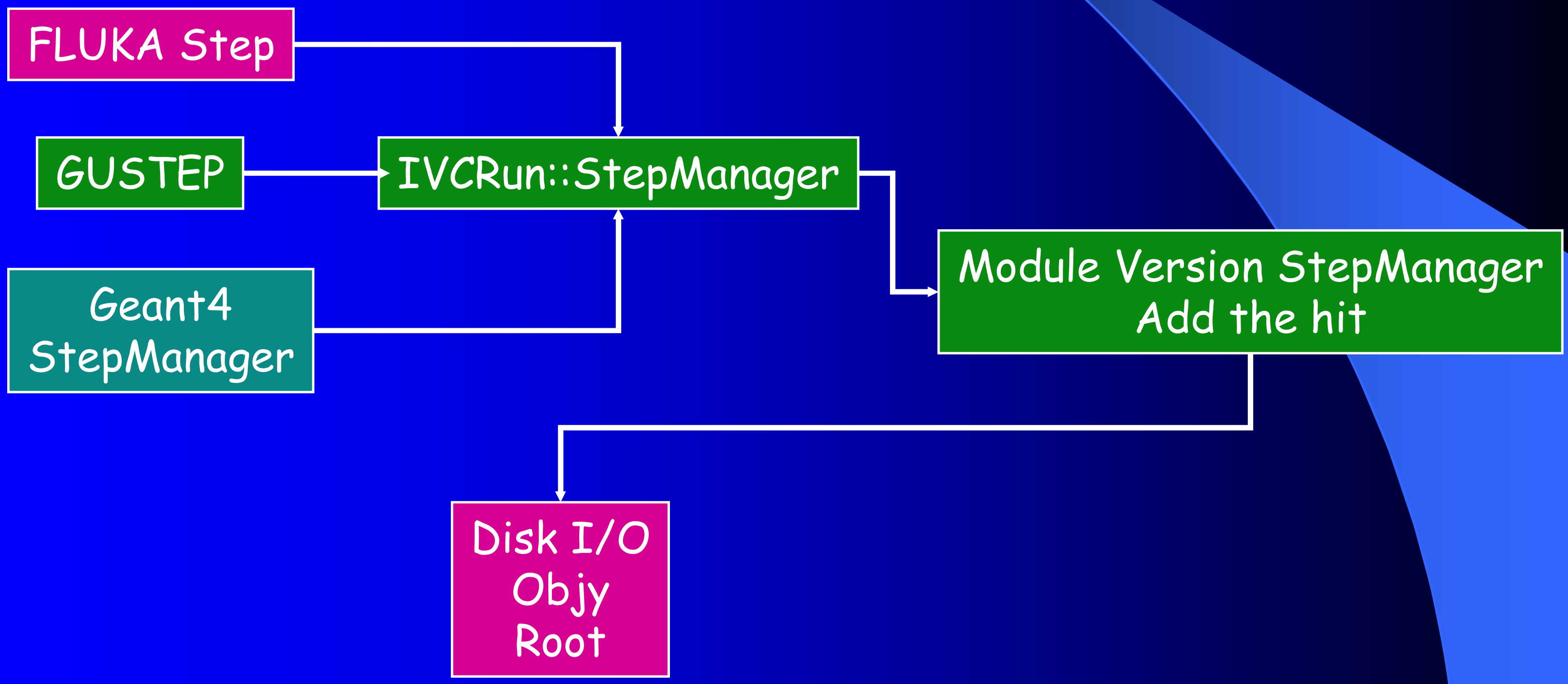
Processing Flow



Traditional GEANT3 Stepping Schema



VMC Stepping Schema



The Interface to GEANT3

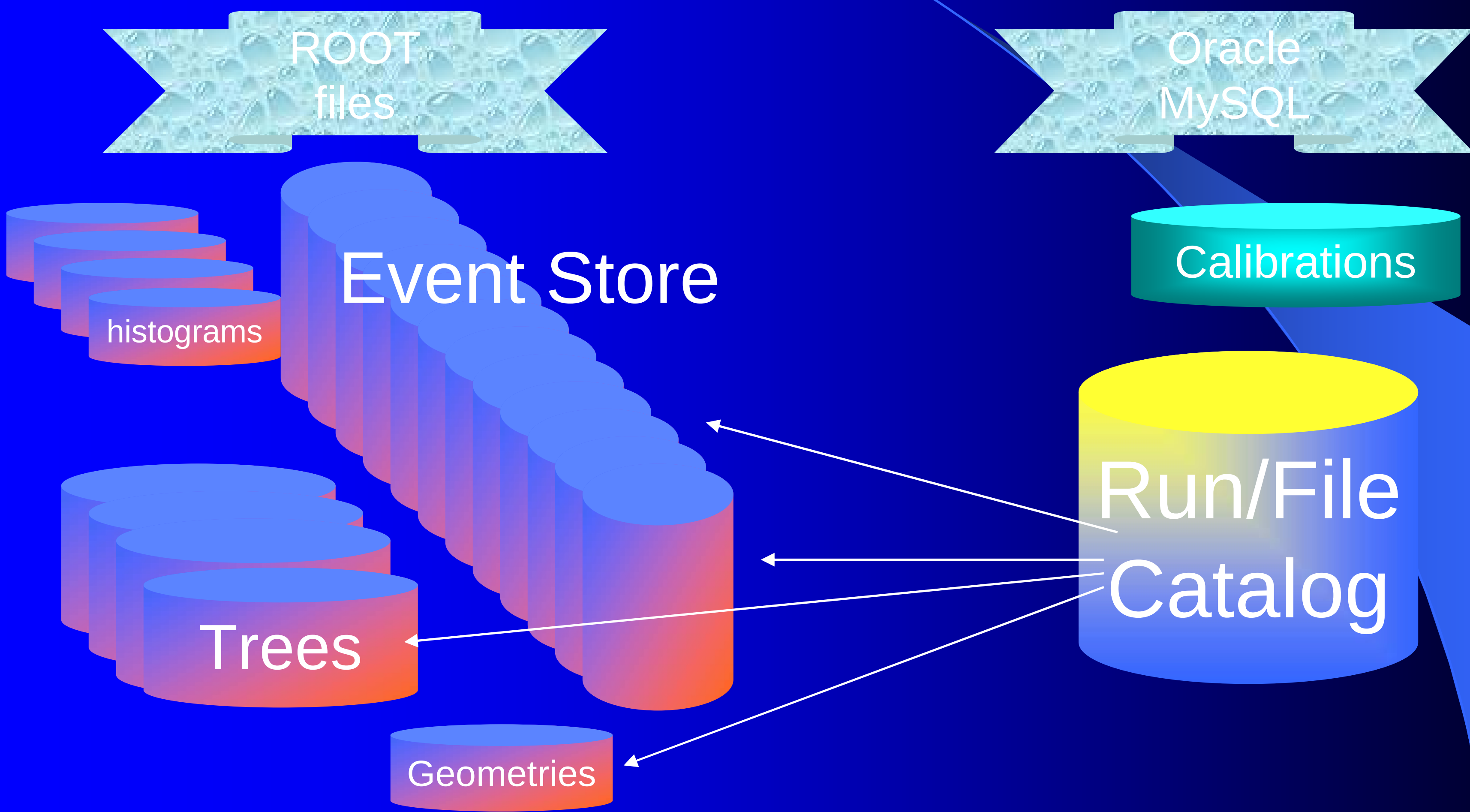
- The class ***TGeant3*** provide the interface between ROOT and GEANT3
- Geant3 subroutines are addressed as methods of an object of the class ***TGeant3***
- The *TGeometry* class describes the geometry of a detector in the GEANT3 style description
- A Geometry object consist of the following linked lists:
  - the TMaterial list (material definition only).
  - the TRotmatrix list (Rotation matrices definition only).
  - the TShape list (volume definition only). the TNode list assembling all detector elements.

Calibration Tasks

- Sub-detector level
  - requires only the sub-detector data and the general event information, for a sub-sample of events.
  - It requires a deep knowledge of the sub-detector, and it is therefore strongly dependent on the group that is responsible for the sub-detector construction, installation and operation.
  - It could be performed at the *Tier* sites where sub-detector groups are active.
  - The database will be accessible by the reconstruction program via one server

- Global calibration
  - Requires only the sub-detector data and the general event information, for a sub-sample of events.
  - Performed at the Tier-0 site by the offline team.
  - The resulting calibration parameters are stored into a database common to all sub-detectors.
  - The *calibration* database will be subsequently accessed through a ROOT interface.

Data Access: ROOT + RDBMS Model



Examples of RDBMS in HEP

- RHIC (started last summer)
  - STAR 100 TB/year + MySQL
  - PHENIX 100 TB/year + Objy
  - PHOBOS 50 TB/year + Oracle
- FNAL (starting this year)
  - CDF 200 TB/year + Oracle
- DESY
  - H1 moving from BOS to Root for DSTs and microDSTs 30 TB/year DSTs + Oracle
- GSI
  - HADES Root everywhere + Oracle
- SLAC
  - BABAR >5 TB microDSTs, upgrades under way + Objy
- CERN
  - NA49 > 1 TB microDSTs + MySQL
  - ALICE + MySQL
  - AMS Root + Oracle

Central Database

- Oracle RDBMS

- Advantages
  - very stable and reliable
  - support for transaction processing
  - built-in procedural language
  - triggers
  - support for complex data types and BLOBs
  - support for VLDB (very large databases), e.g. data partitioning
  - 7 × 24 availability (on-line backup, etc.)

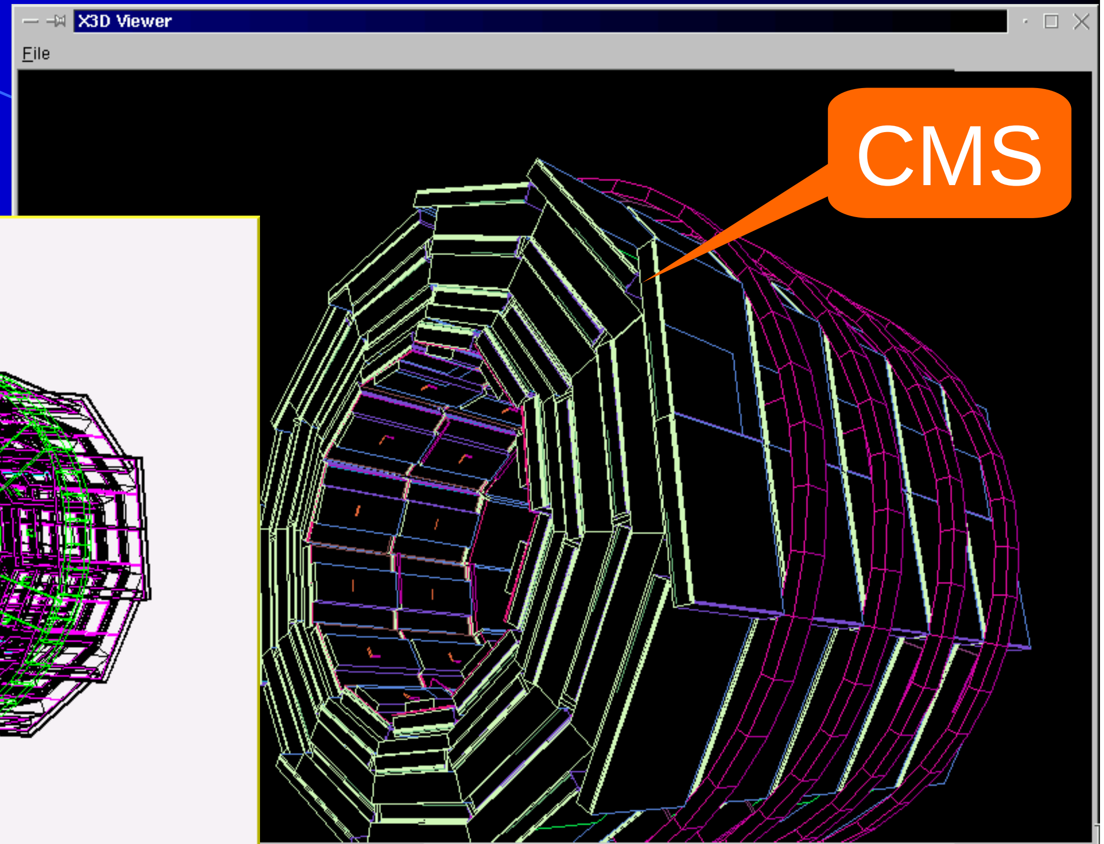
- Disadvantages
  - quite expensive
  - complex and difficult to administer

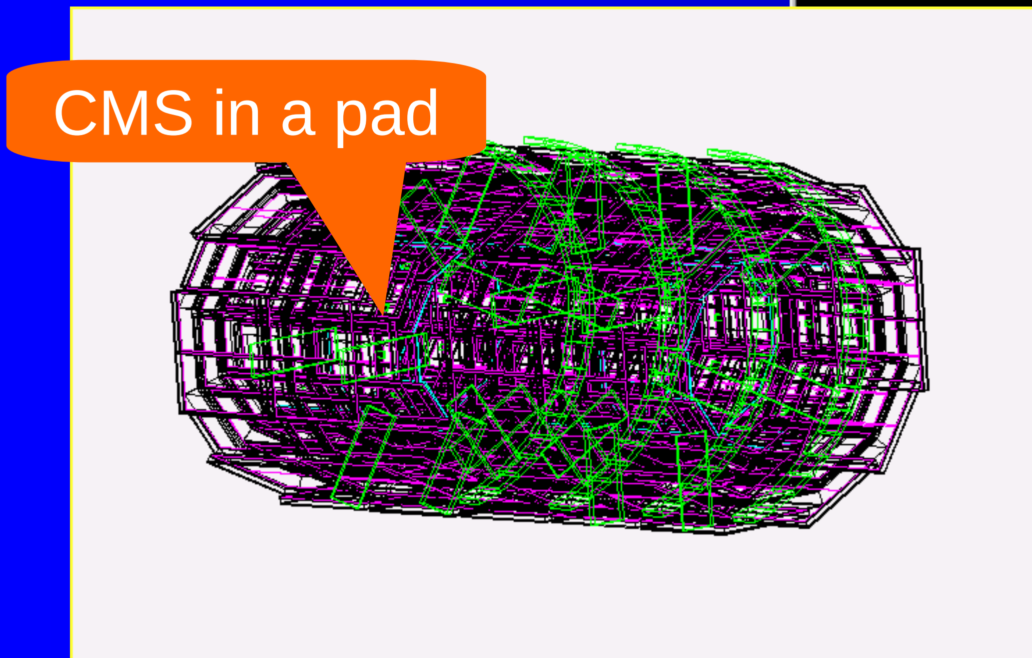
- PostgreSQL / MySQL

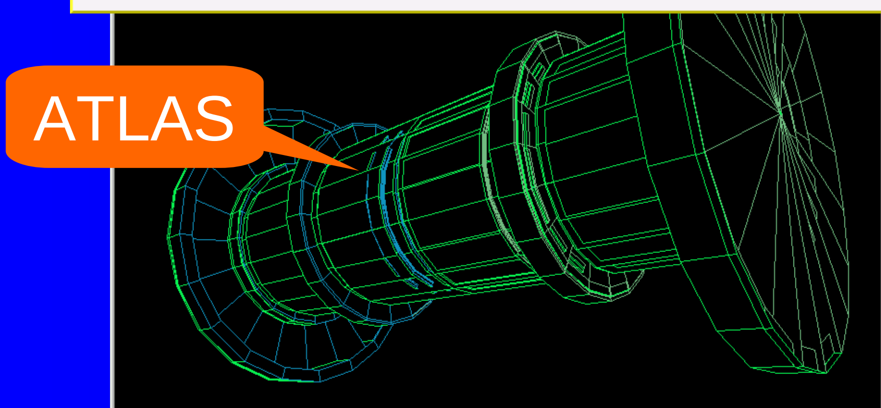
- Advantages
  - free of charge
  - quite easy to administer
  - stable enough
  - support for transaction processing
  - built-in procedural languages
  - triggers
  - support for complex data types and BLOB objects

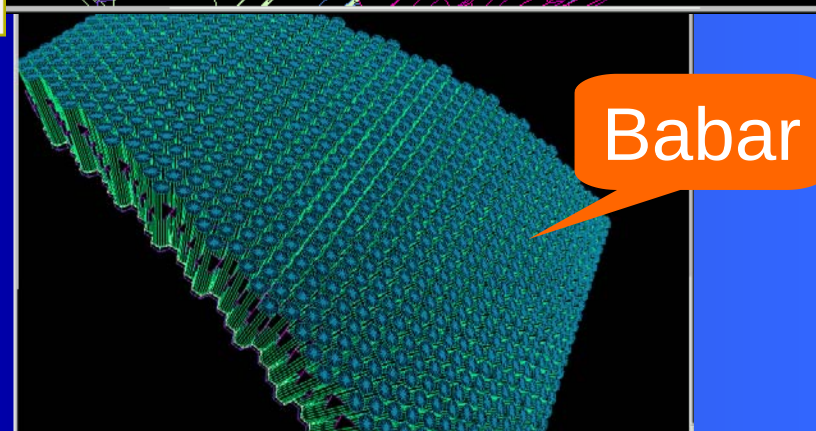
- Disadvantages
  - not *very* fast (but fast enough for this particular application)
  - no support for distributed processing (data replication, etc.)
  - no support for heterogeneous systems
  - no support for VLDB
  - no 7 × 24 availability

3-D views



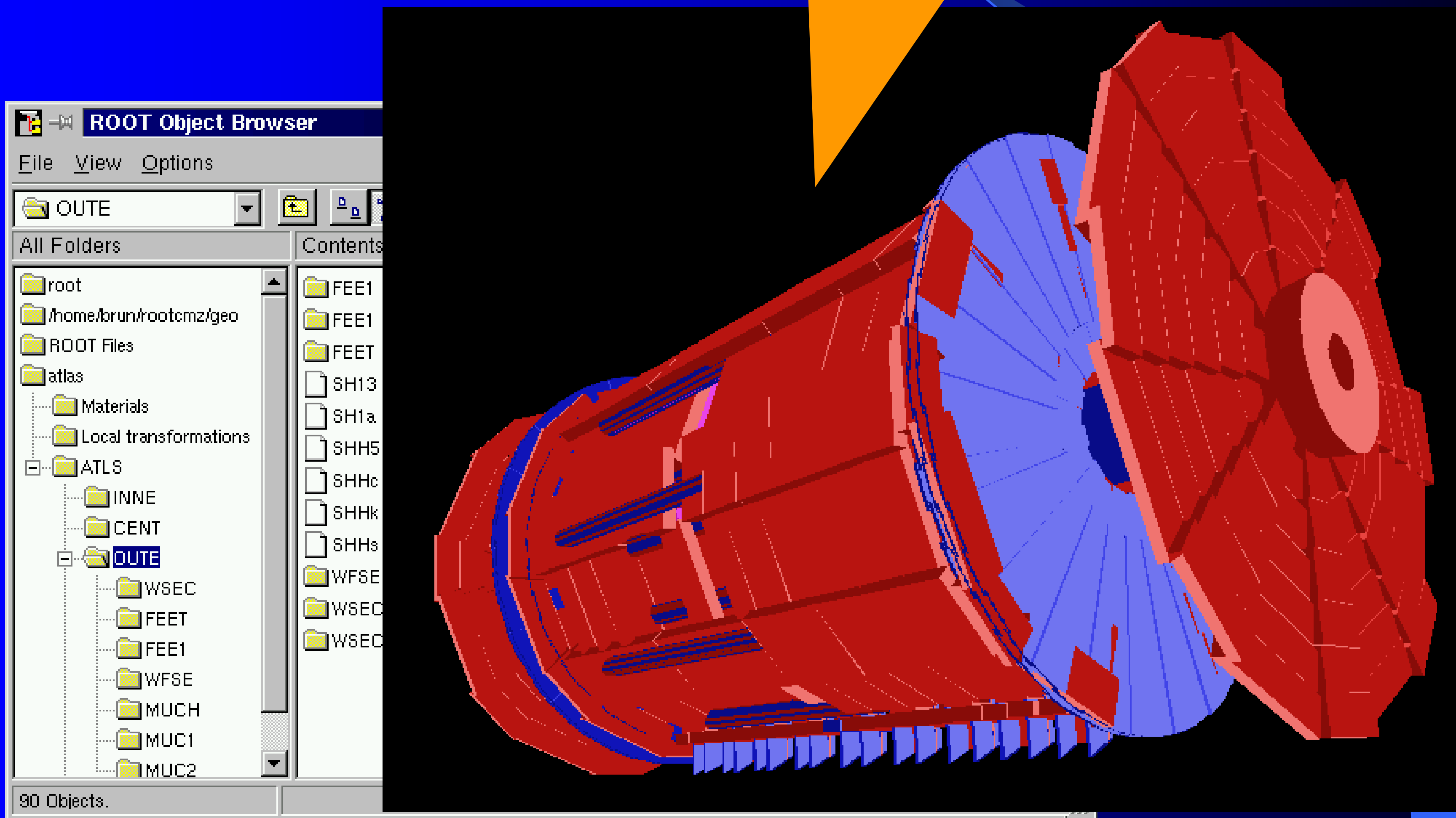




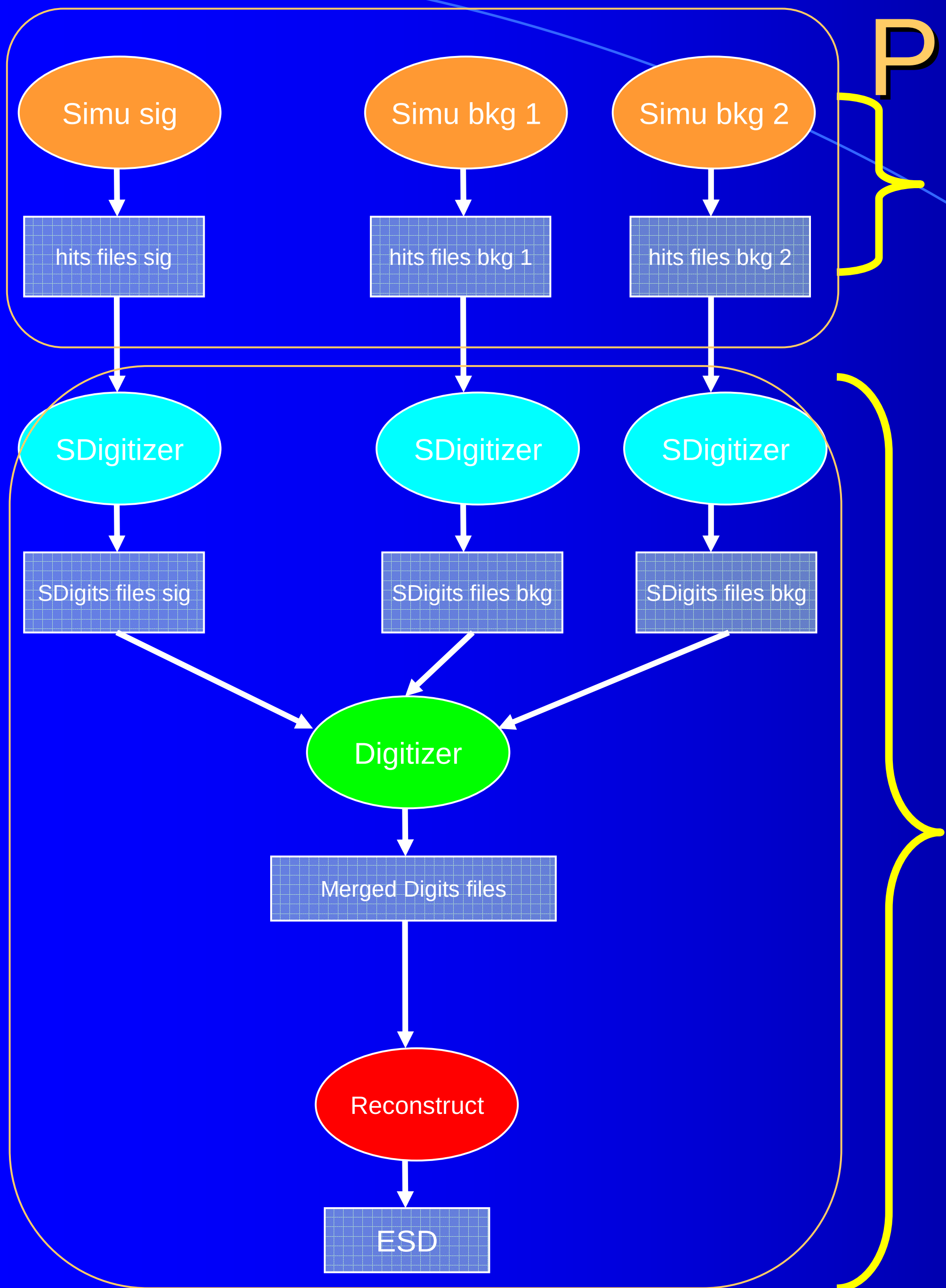


TGeo example: Atlas

29 million nodes



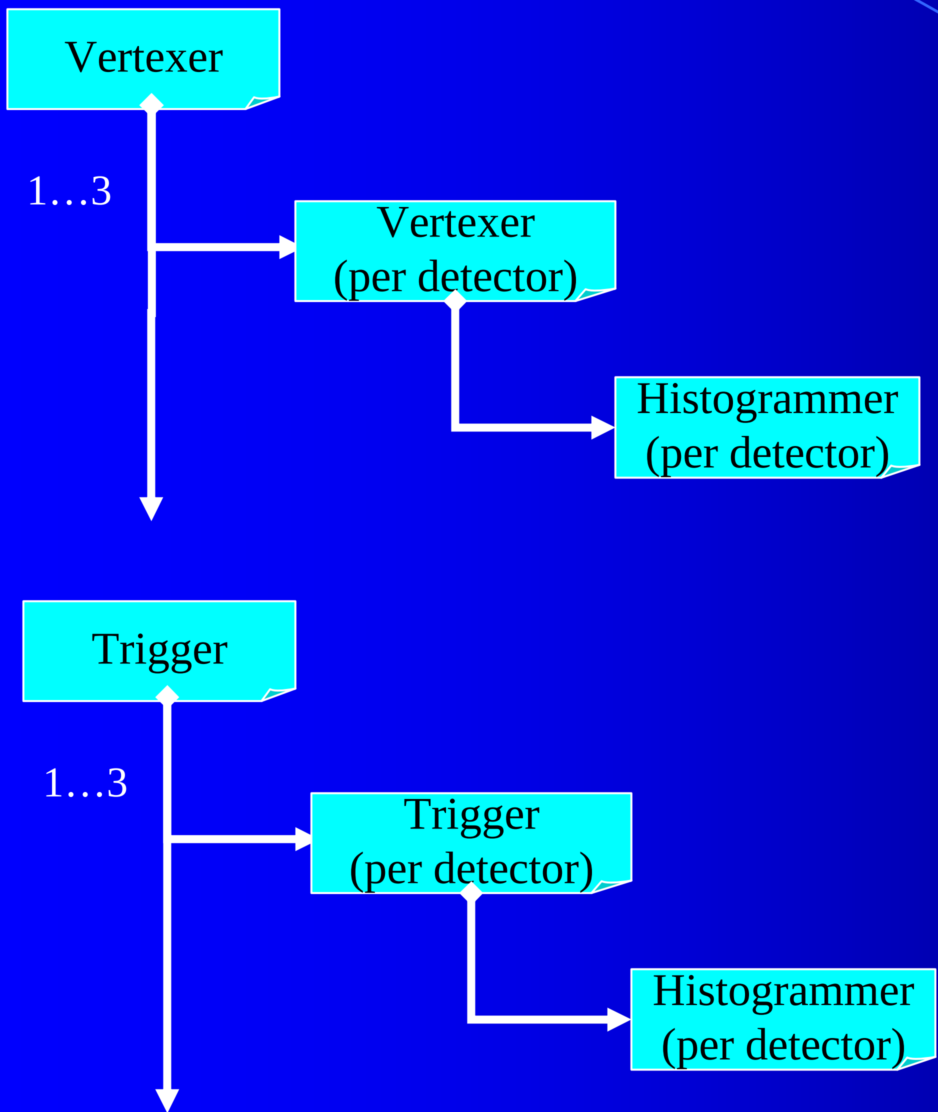
Processing Flow



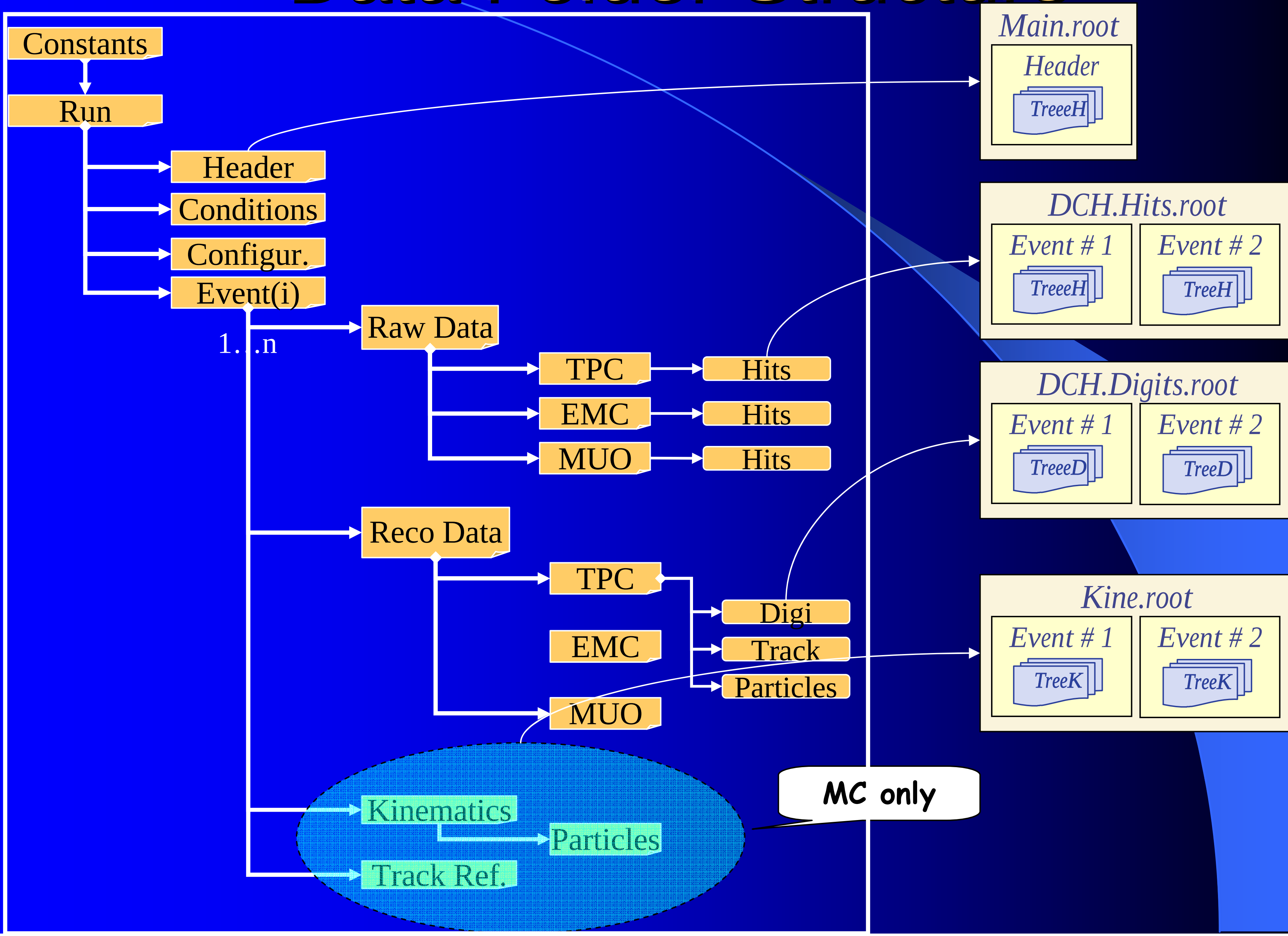
GRID

- A specific interface to EDG is being developed at CERN (AliEn)
- It requires API EDG SE
- Technology needs to be improoved
- Talks in progress with Carminati for a possible involvement of the Lecce group

Folders Type: Tasks



Data Folder Structure

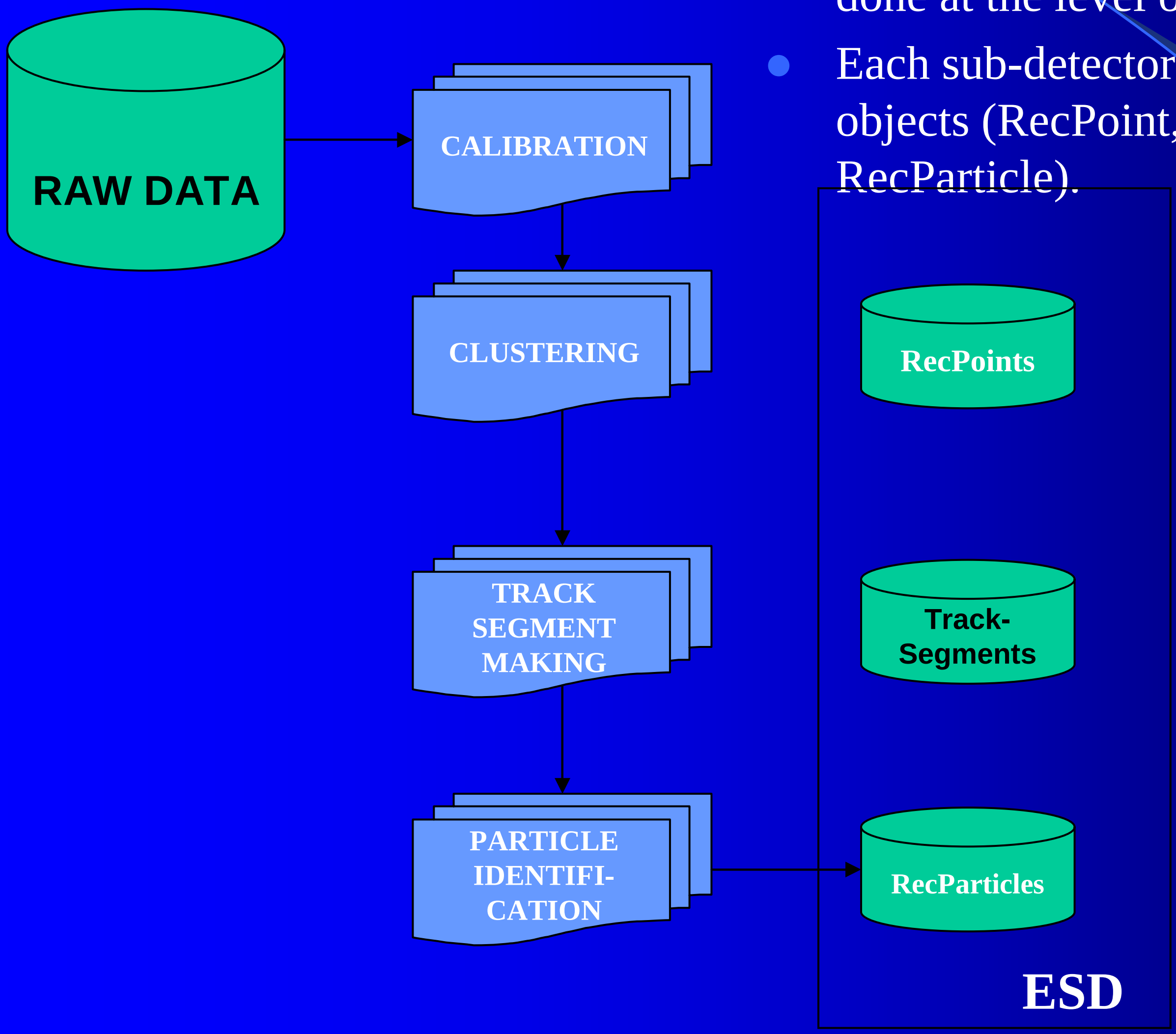


Data Model

- *ESD (Event Summary Data)*
  - contain the reconstructed tracks (for example, track pt, particle Id, pseudorapidity and phi, and the like), the covariance matrix of the tacks, the list of track segments making a track etc…
  -
- *AOD (Analysis Object Data)*
  - contain information on the event that will facilitate the analysis (for example, centrality, multiplicity, number of electron/positrons, number of high pt particles, and the like).
- *Tag* objects
  - identify the event by its physics signature (for example, a Higgs electromagnetic decay and the like) and is much smaller than the other objects. Tag data would likely be stored into a database and be used as the source for the event selection.
- *DPD (Derived Physics Data)*
  - are constructed from the physics analysis of AOD and Tag objects.
  - They will be specific to the selected type of physics analysis (ex: mu->e gamma, mu->e e e)
  - Typically consist of histograms or ntuple-like objects.
  - These objects will in general be stored locally on the workstation performing the analysis, thus not add any constraint to the overall data-storage resources

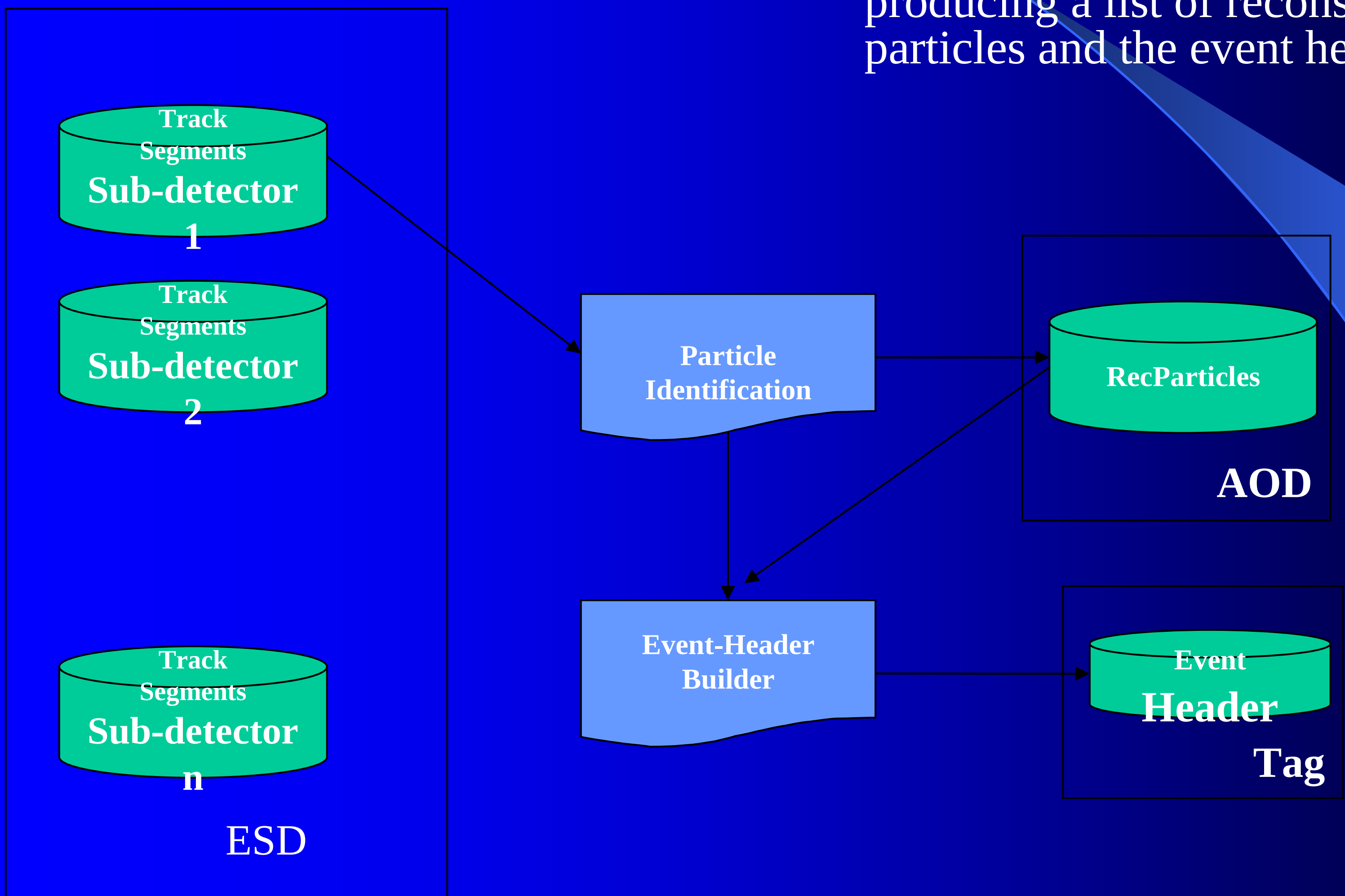
Reconstruction flow and ESD objects

- The various tasks (calibration, clustering, …) are done at the level of the sub-detectors
- Each sub-detector produces a list of reconstructed objects (RecPoint, TrackSegment, and RecParticle).



Reconstruction flow, AOD and Tag

- Task is accomplished with the cooperation of the sub-detectors, producing a list of reconstructed particles and the event header.



Montecarlo Processing Flow