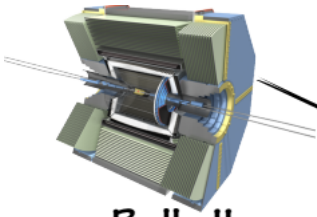
Belle II Analysis Model
in a nutshell

Guglielmo De Nardo

Università di Napoli Federico II and INFN

Jennifer Consortium General Meeting, Rome, 9-11 June 2015

Data flow



Belle II Detector

Data processed @ "raw data center"

DAQ unit
DAQ unit
DAQ unit

Online disk
(Sequential root)

Offline storage
(Sequential root
→ raw data in root : merge)
300kB / event

first process

Reconstruction

mDST 40kB / event

MC produced on the world-wide distributed computing

Event generation

Detector simulation

Digitization

No intermediate stage is saved

~4 sec/event

Reconstruction

40kB / event mDST

distributed to each region

"coordinated" group skimming

group official skim : mDST, μDST, index

mDST : reconstruction level info.,
μDST : mDST + particle level info,
index : collection of pointers to events

User distributed analysis

"Conceptual" index ← User job → Ntuple

less disk requirement
more chaotic network access

mDST

"another possibility"

User job → Ntuple

μDST

more disk requirements
less network requirements

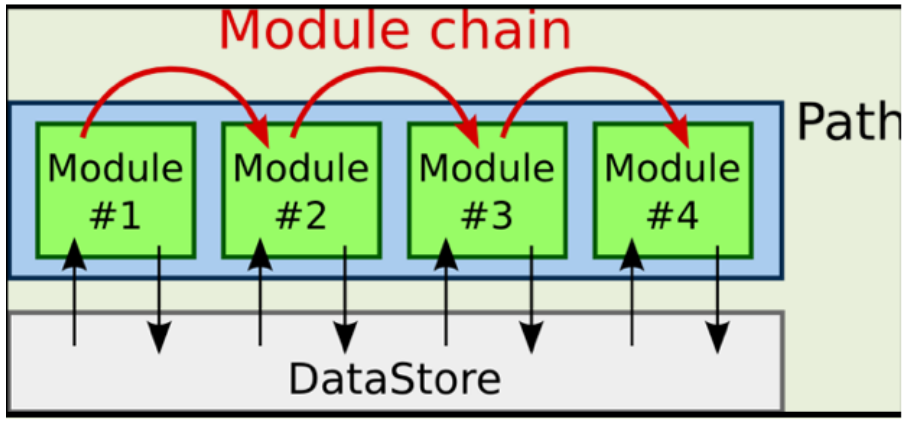
Software Framework

- Simulation, reconstruction and analysis performed by set of modules managed by a framework (basf2)

Algorithms are coded in modules (C++ code)

Modules are put in a path and executed sequentially

A Datastore manages data loaded or created by modules

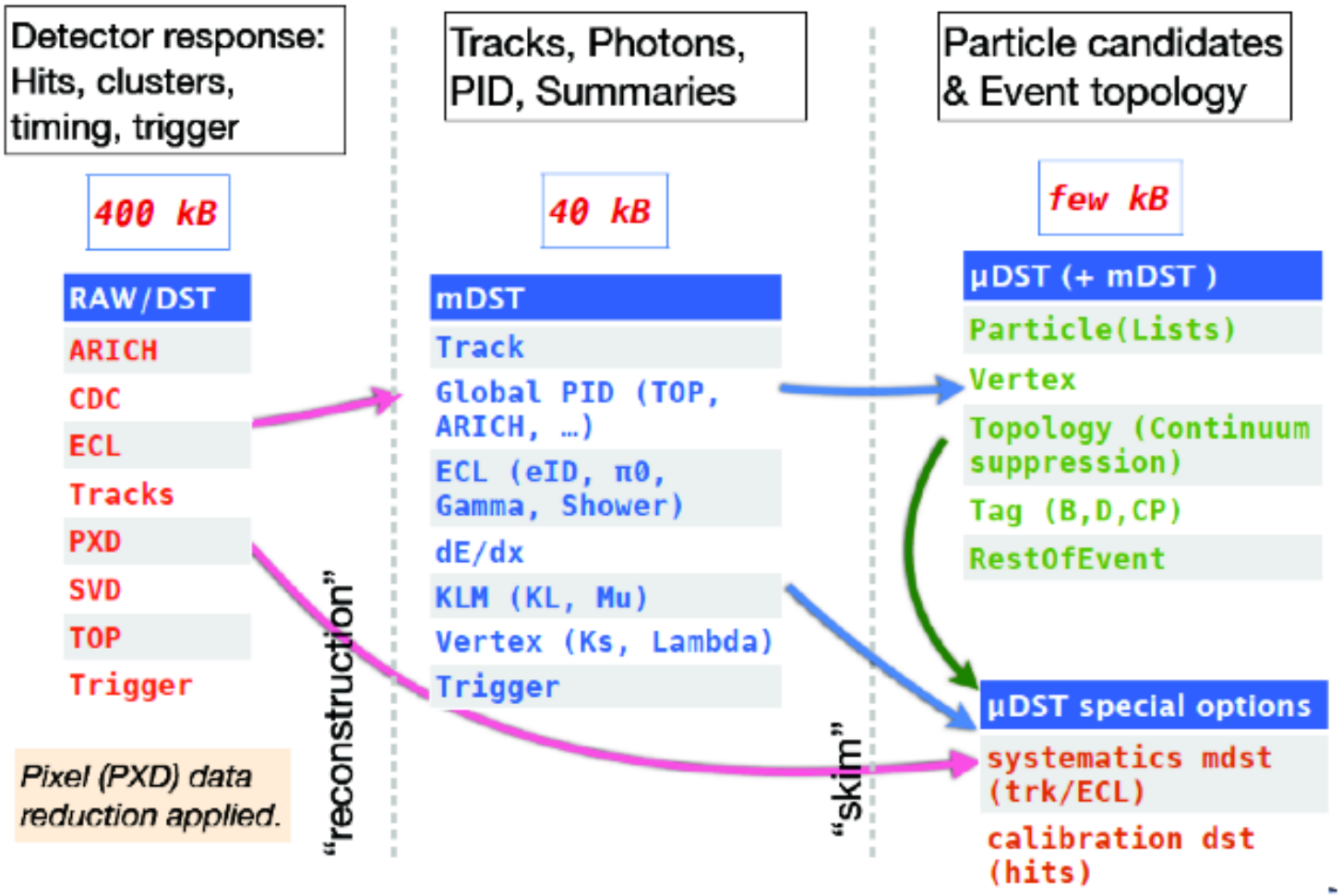


Datastore can handle any class that inherit from a ROOT TObject and has a dictionary
Any part of the datastore can be persisted in ROOT format

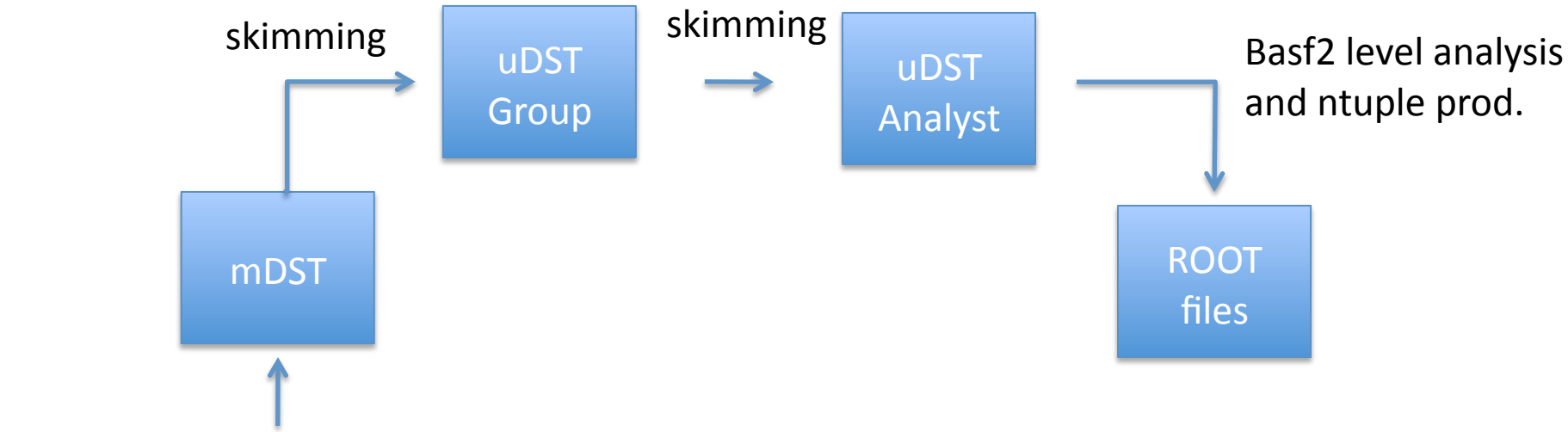
Data Model

- Several layer of data
  - lowest level are
    - Real data:what DAQ brings out from detector (RAW data)
    - Simulation: simulated detector response is saved as SIMHits data (from Geant4) + MC physics generators output
  - Regardless of the origin detector response is represented by “Hits” (subsystem-wise) that are the input of reconstruction algorithms

Reconstructed Data model



“Ideal” analysis model



mDST → uDST : data size is reduced by replacing reconstruction level objects with highest level particle candidates with kinematics/vertexing/PID information

Skimming:
- whole events are rejected, on the basis physics selection requirements
- Additional observables and decay tree candidates augment information and size.

uDST (Physics level data)

Basic objects are particle candidates and their containers o

Based the minimal definition of a physics particle (kinematics+vertexing+PID)

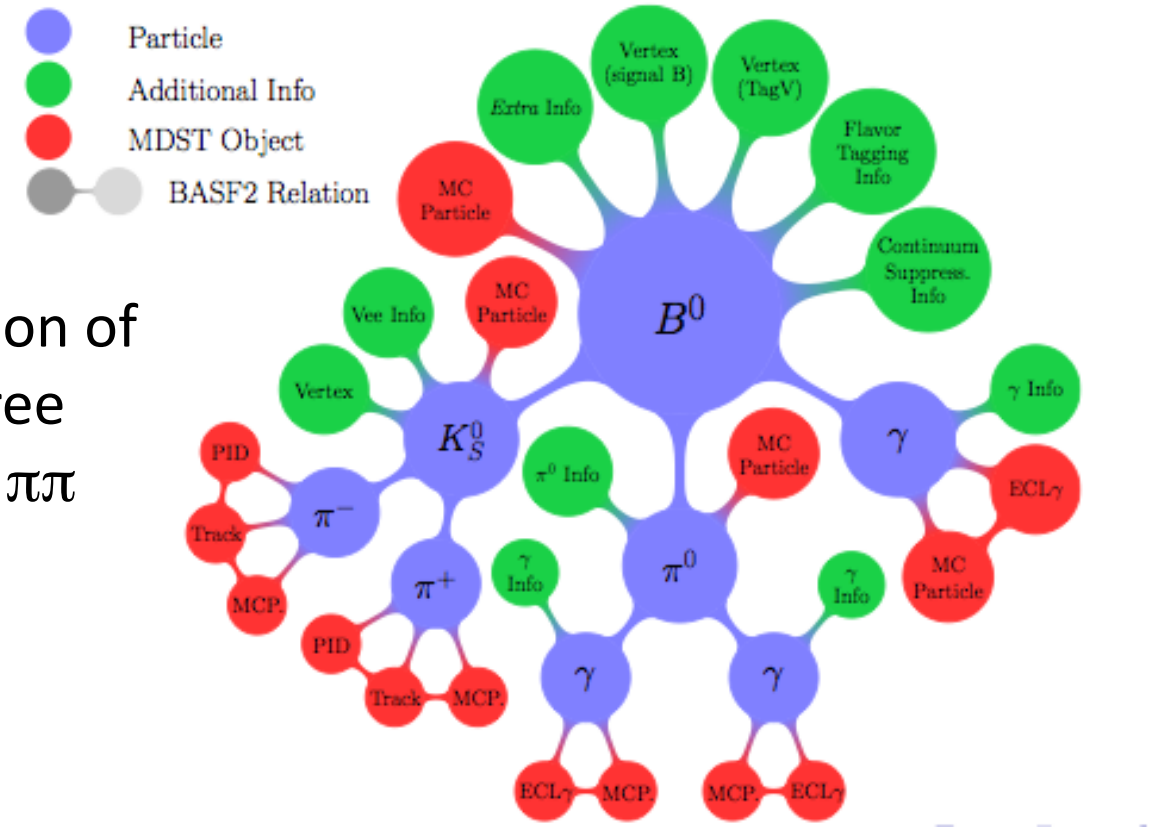
Physics Tools are framework modules manipulating particles

Composition tools make new intermediate particles managing the decay tree combining candidate final states

Selection tools can apply selection algorithms to refine particle lists

Specific tools build additional observables associated to candidates

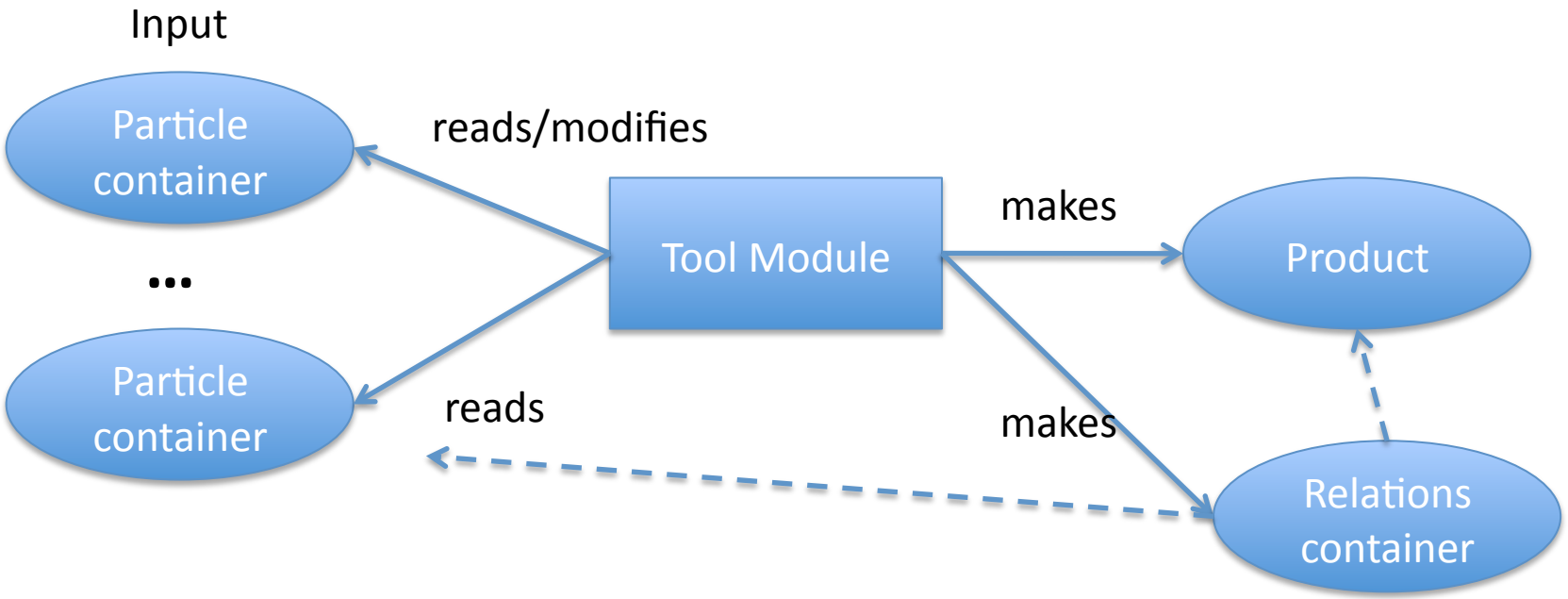
A pictorial representation of a $B^0 \rightarrow K_S \pi^0 \gamma$ decay tree with $\pi^0 \rightarrow \gamma\gamma$ and $K_S \rightarrow \pi\pi$



Additional observable like MC Truth matthcing, vertexing, B tagging, continuum suppression, Rest of Event information are added to the B candidate and some of the particle candidates in the decay tree

Abstract Analysis Tools scheme

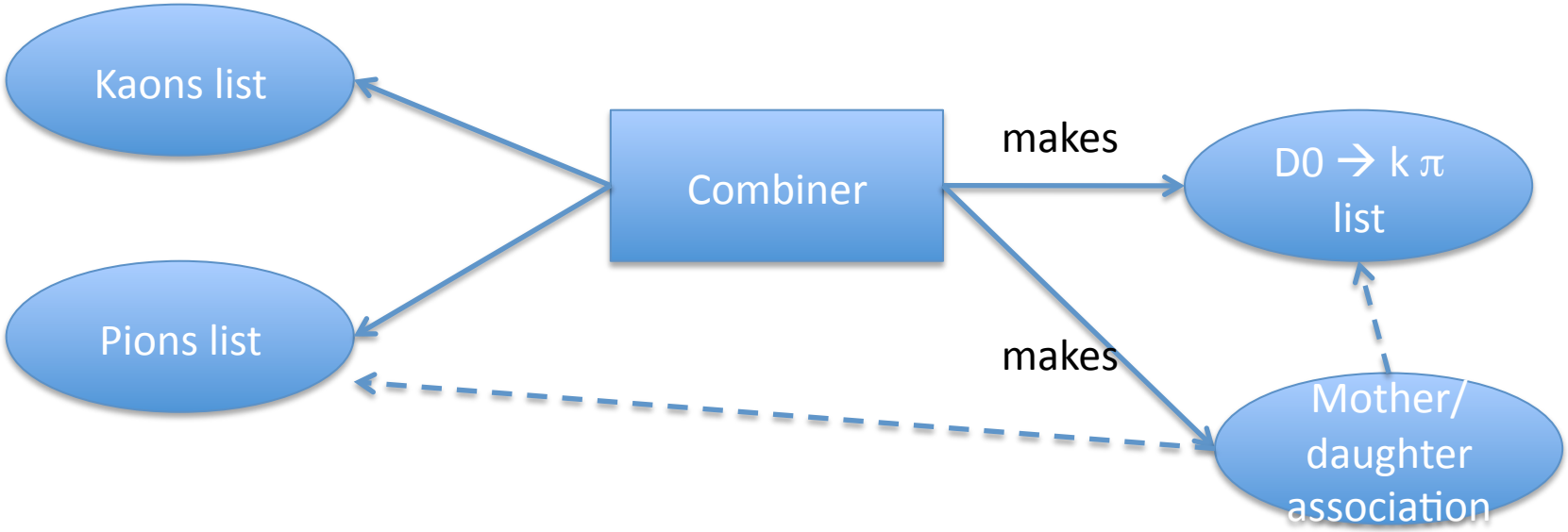
Several specialized modules available to perform single well defined actions on input Particle candidates producing output candidates



A tool may use one or more input container of Particles
Produce as output a product.
A product may be a Particle container or additional data container (continuum suppression, rest of event data, …) attached to an input candidate

Concrete implementation example

From all kaons and pions (made by other tools) make D0 → k p candidates that satisfy Some selection criteria



The combiner tool has to be configured with selection criteria in addition to the input and output container names

User interface (Python)

- A set of utility functions and conventions have been developed to standardize most common “analysis pattern” followed by analysts
  - Standard selection of final states (PID)
  - Calculation of well known observables
  - Vertexing, B tagging, full event reconstruction…
  - Common output ntuple format
- Goal is to minimize duplication of efforts and let the analysts focus on physics
- Brave analyst can still write C++ code in her own analysis module
  - The default approach would instead be to configure the application steering file for data selection without re-compiling basf2

Sample Analysis “script”

This snippet let the analyst write a root file with a tree with kinematics, PID, and MC for events where D* → D π, D → K π candidates are reconstructed with a loose selection Criteria.

```
inputMdstList(inputfiles)
stdLoosePi()
stdLooseK()
reconstructDecay('D0:kpi -> K-:loose pi+:loose', '1.8 < M < 1.9')
reconstructDecay('D*+ -> D0:kpi pi+:all', '0.0 < Q < 0.020 and 2.5 < useCMSFrame(p) < 5.5’)
matchMCTruth('D*+')
toolsDST = ['EventMetaData', '^D*+']
toolsDST += ['InvMass', '^D*+ -> ^D0 pi+']
toolsDST += ['CMSKinematics', '^D*+']
toolsDST += ['PID', 'D*+ -> [D0 -> ^K- ^pi+] ^pi+']
toolsDST += ['Track', 'D*+ -> [D0 -> ^K- ^pi+] ^pi+']
toolsDST += ['MCTruth', '^D*+ -> ^D0 ^pi+']
ntupleFile('B2A301-Dstar2D0Pi-Reconstruction.root')
ntupleTree('dsttree', 'D*+', toolsDST)
process(analysis_main)
```

Conclusions

- General strategy Belle II analysis model is well defined
- Most of implementations details are in place and being used (→tested)
  - Some others in development (interface with detector conditions DB, for example)
- Details of critical parts like skimming, tuning mDST/uDST content will evolve with time
  - We are going to verify how they scale with large datasets and group/analysts “usage patterns”.