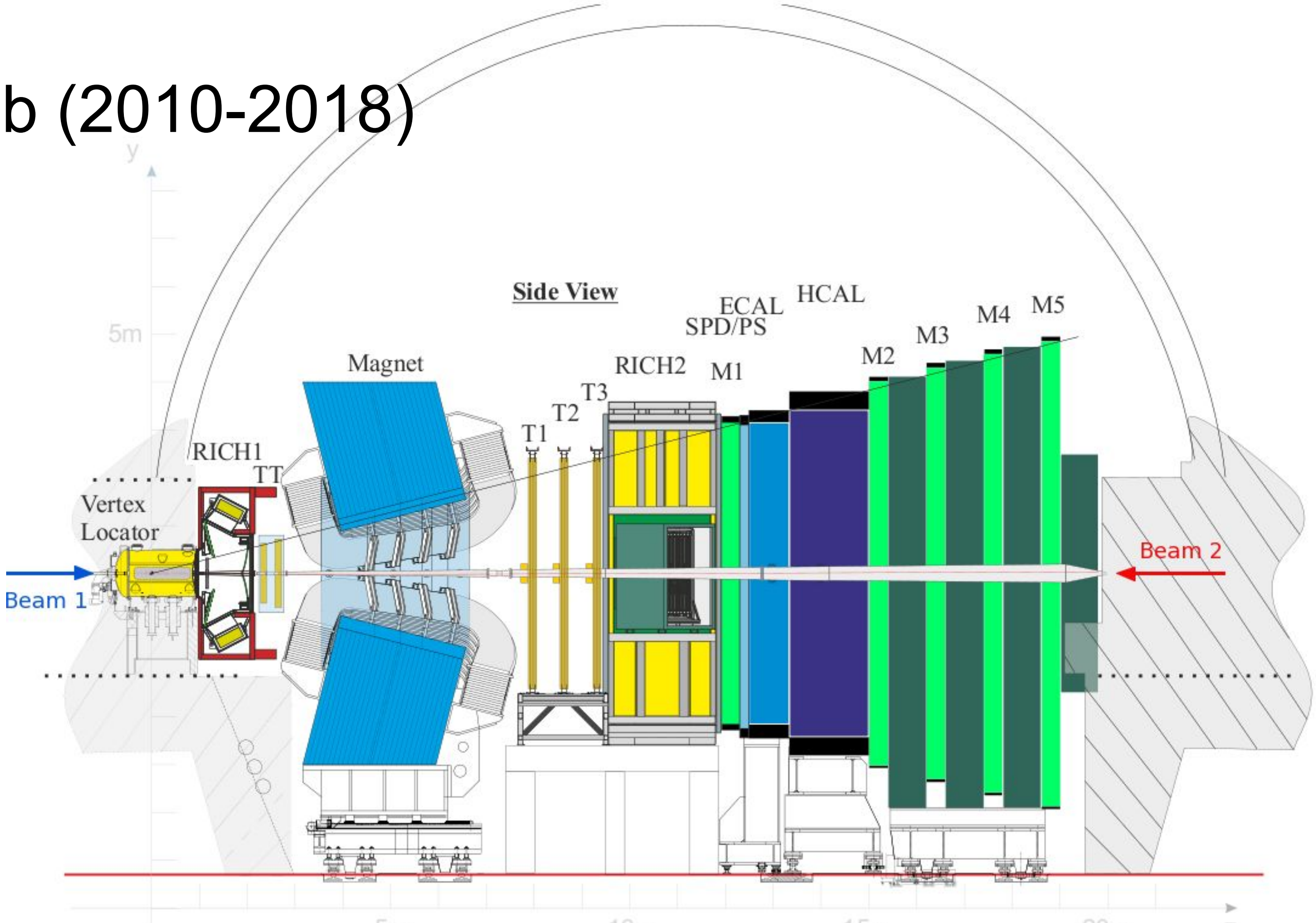
LHCb Data flow

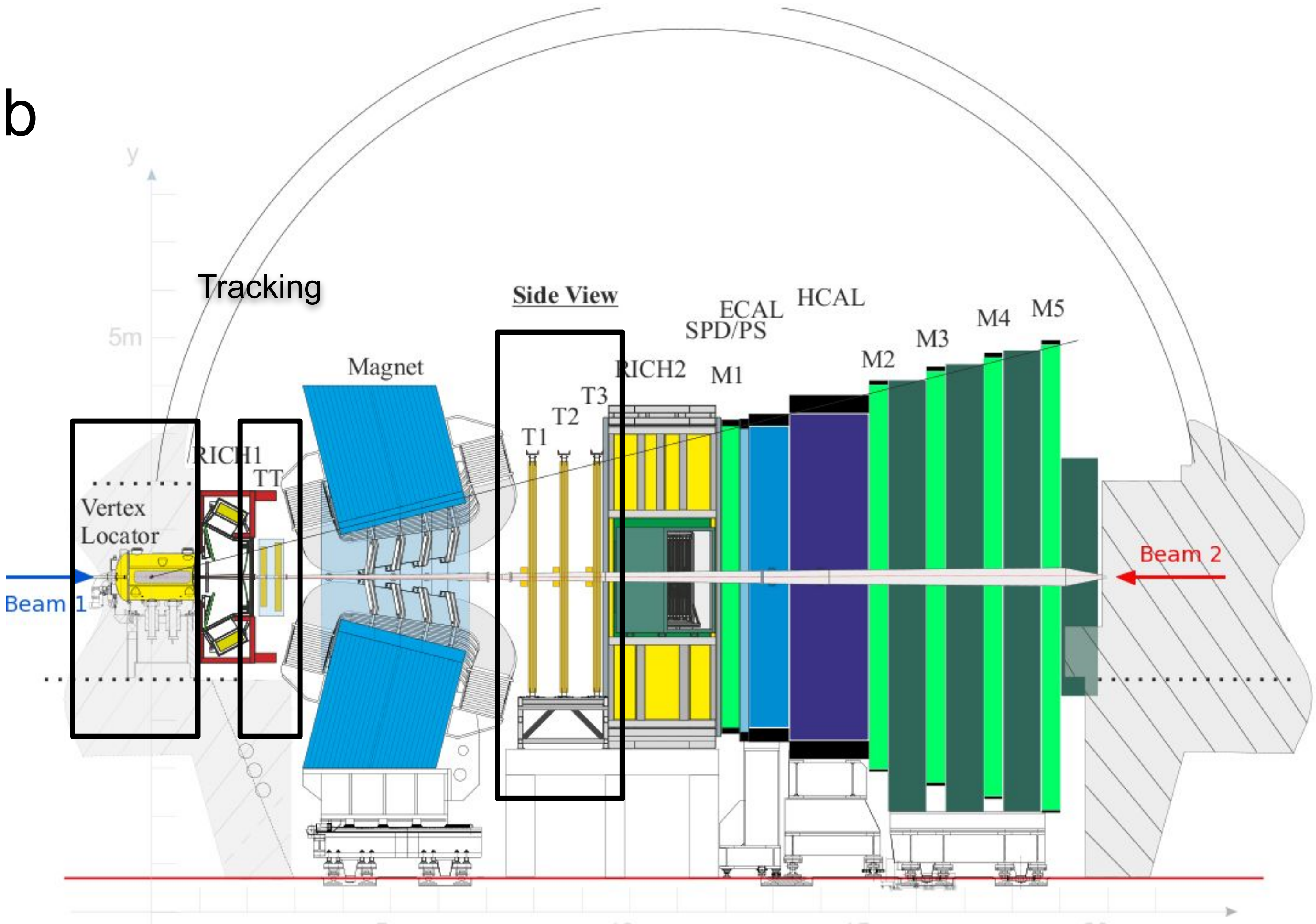
Valeriia (Lera) Lukashenko

LHCb Starterkit 2020

LHCb (2010-2018)

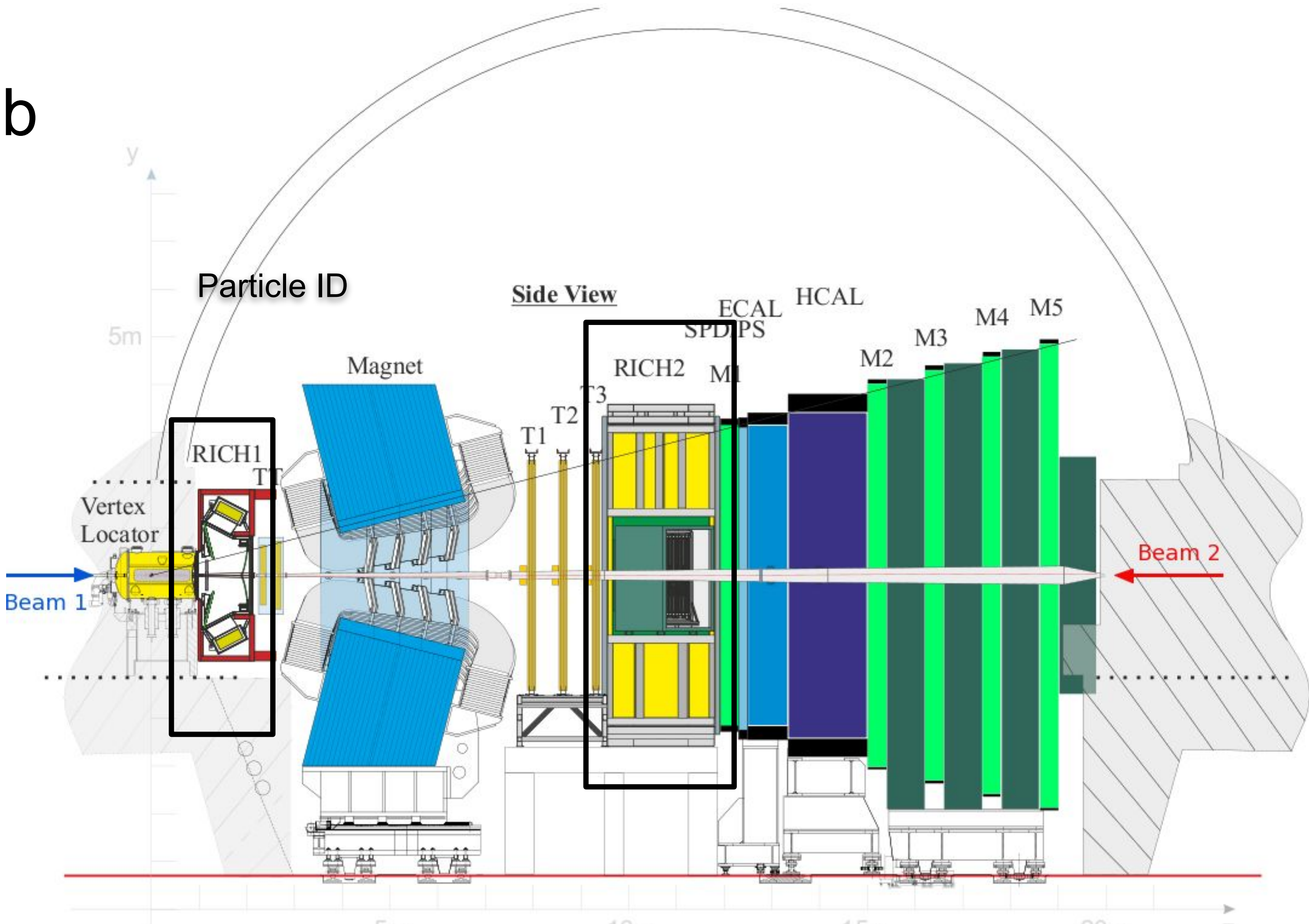


LHCb

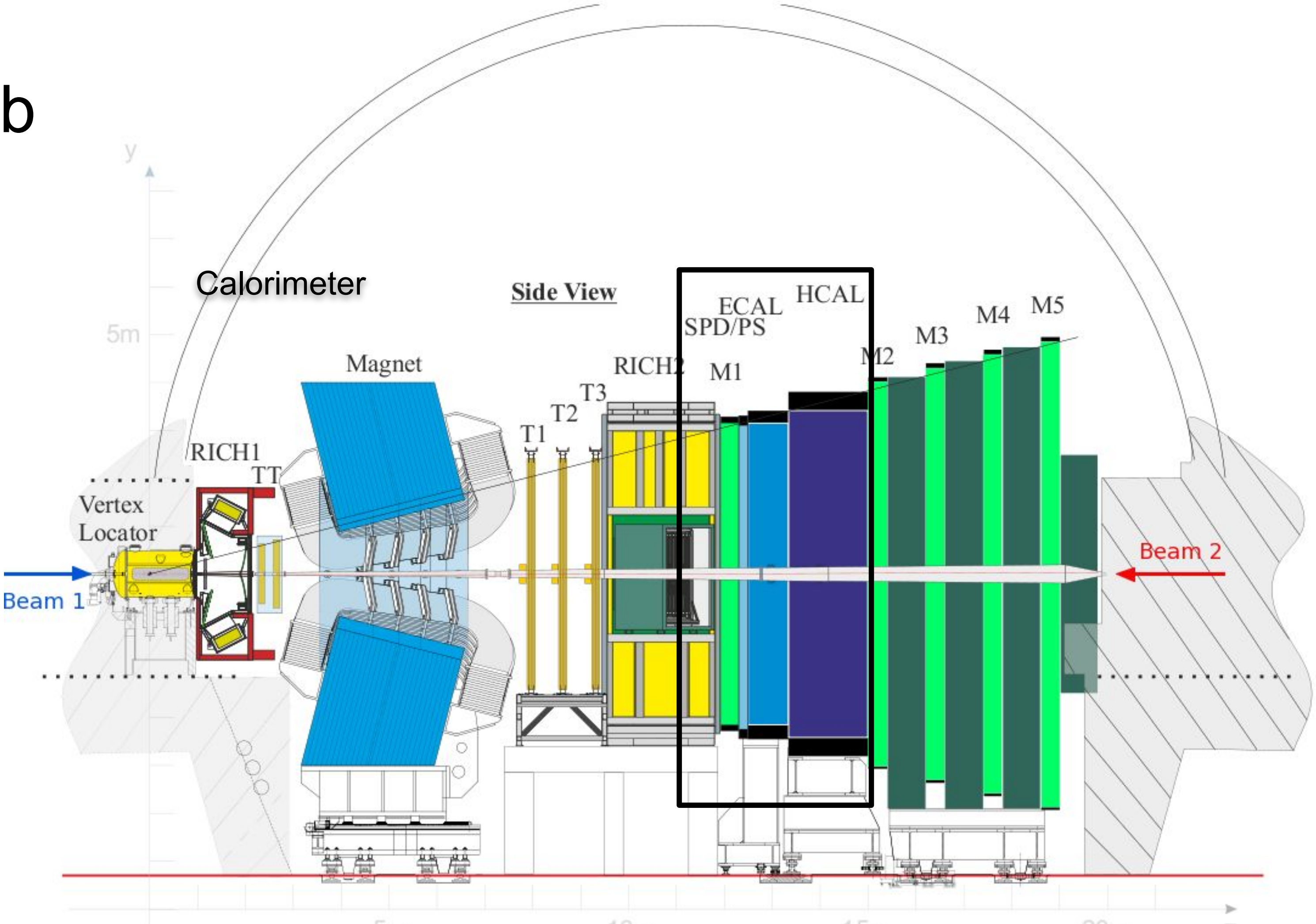


Tracking
Side View
y
5m
Magnet
RICH1
TT
Vertex Locator
T1
T2
T3
RICH2
SPD/PS
ECAL
HCAL
M1
M2
M3
M4
M5
Beam 1
Beam 2

LHCb


Particle ID
Side View
5m
y
Magnet
RICH1
TT
Vertex
Locator
T1
T2
T3
RICH2
ECAL
SPD/PS
M1
HCAL
M2
M3
M4
M5
Beam 1
Beam 2

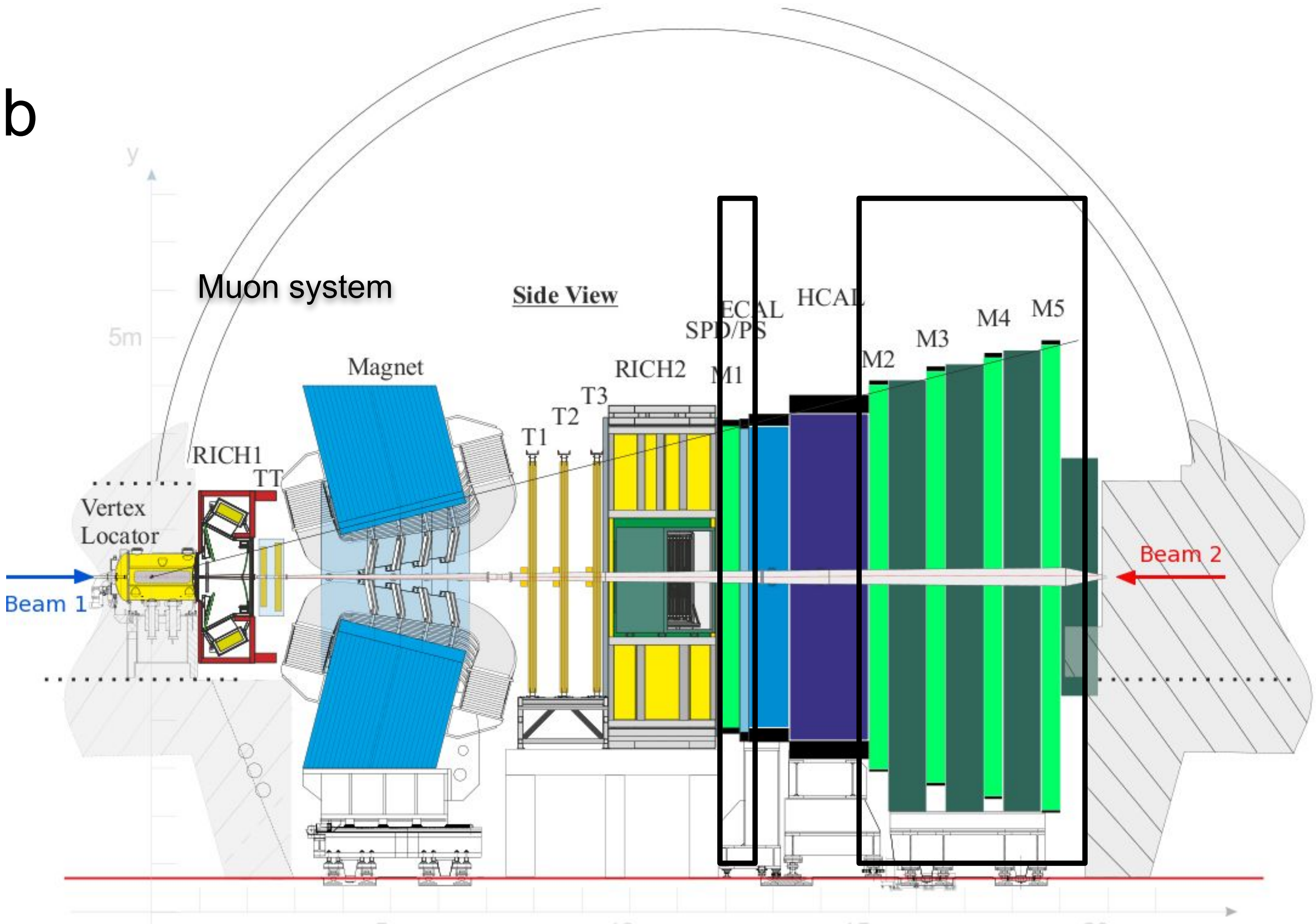
LHCb



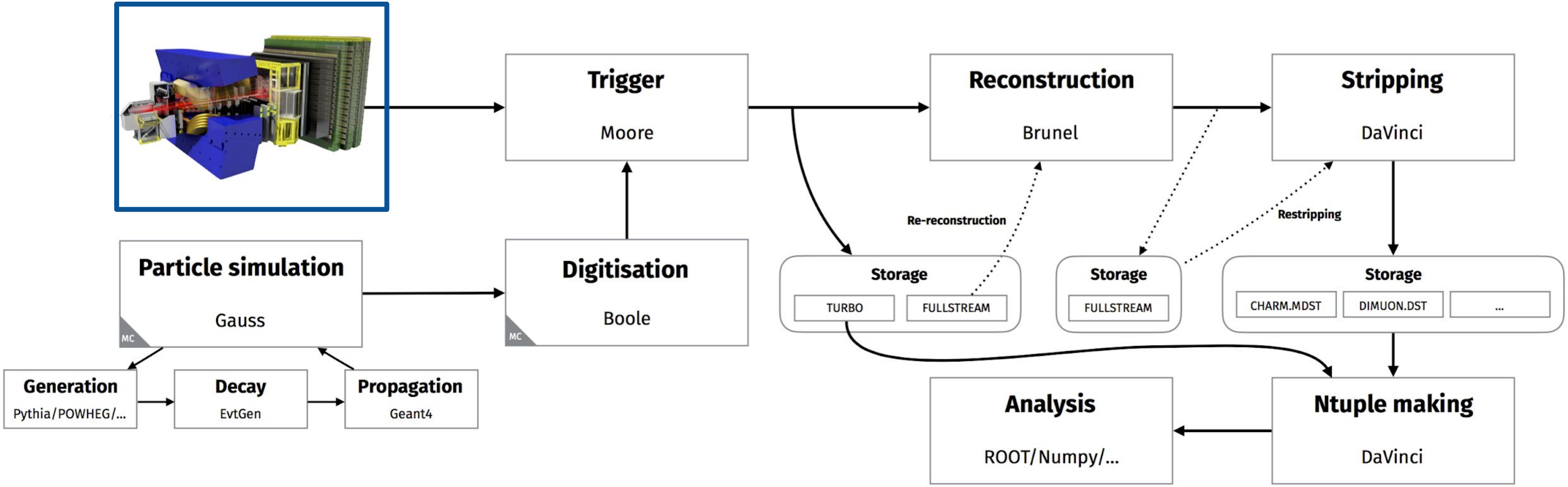
Calorimeter
Side View
y
5m
Magnet
RICH1
TT
Vertex
Locator
T1
T2
T3
RICH2
ECAL
SPD/PS
HCAL
M1
M2
M3
M4
M5
Beam 1
Beam 2

LHCb

Muon system



LHCb Data Flow Run II (2015-2018)



➔ LHC: 40 MHz collision rate ~ 1TB/s of information

➔ Our resources are limited

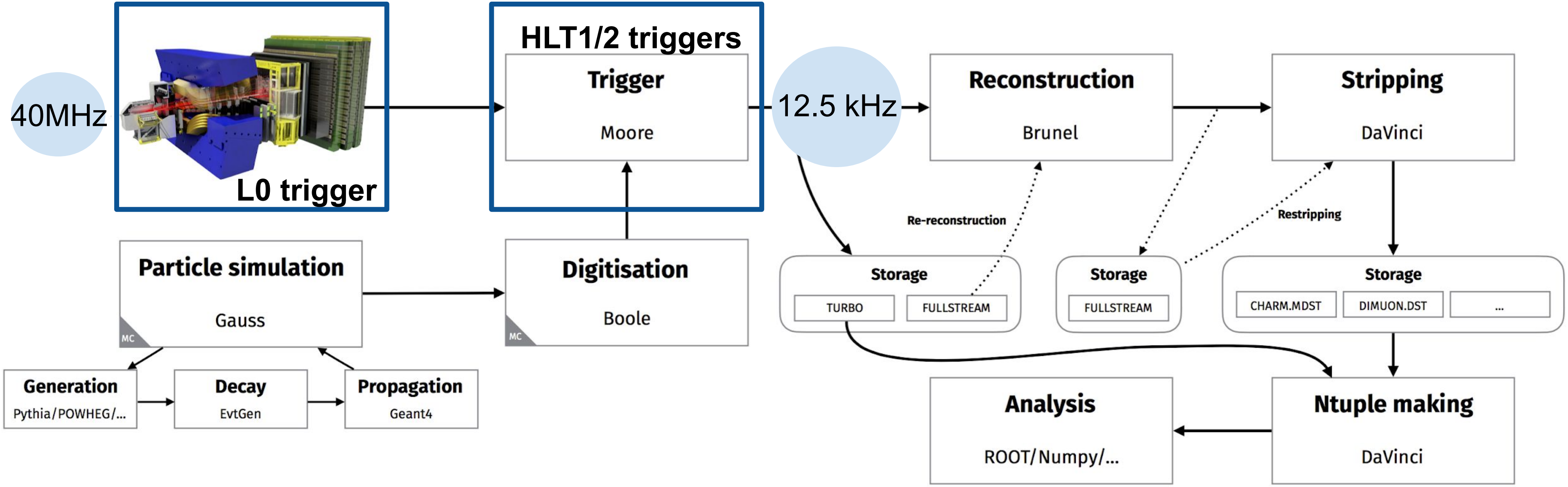
LHCb Data flow

We want to save only the interesting stuff

We want to make decision quickly and accurately

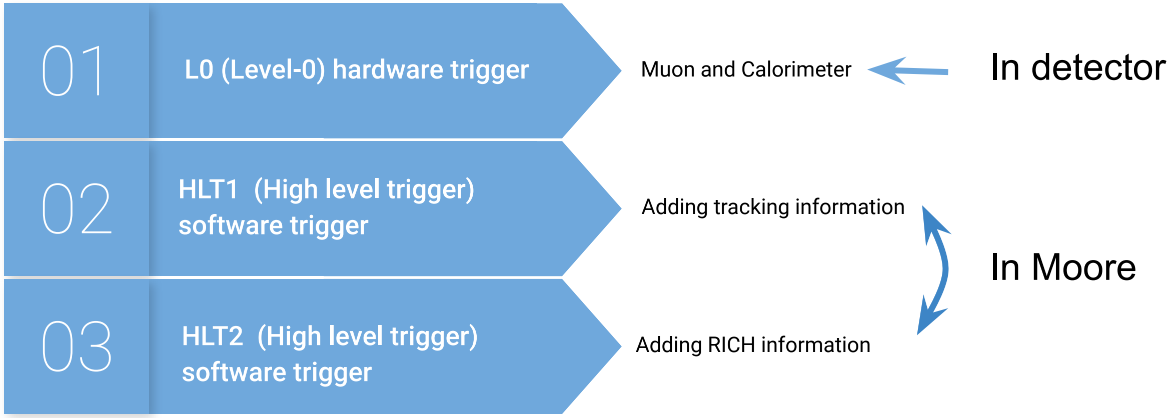
We can use limited resources

LHCb Data Flow Run II. Stage 1: trigger



Trigger stage of the data flow is also called online reconstruction.

LHCb Data Flow Run II. Stage 1: trigger

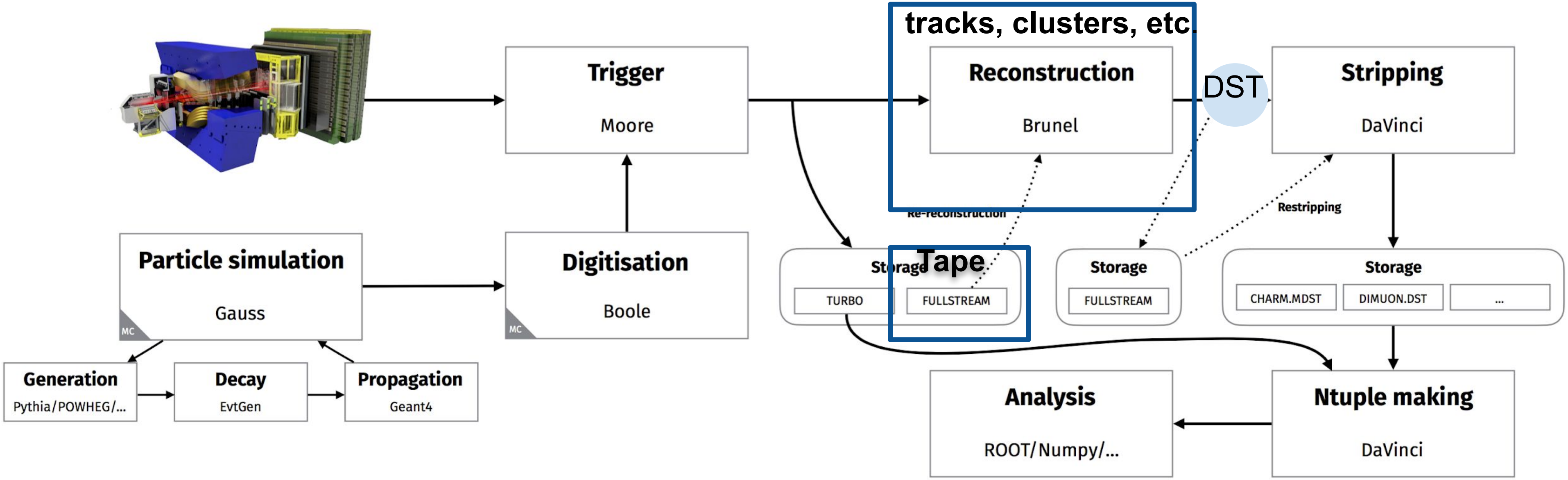


Moore project on gitlab

Fast trigger often means worse resolution.*
Therefore, we might need extra offline reconstruction.

*Spoiler alert: In Run II the online reconstruction is actually precise enough to skip offline reconstruction. About it in a few slides.

LHCb Data Flow Run II. Stage 2: reconstruction



Trigger stage of the data flow is also called offline reconstruction.

Brunel on gitlab

Rec on gitlab

LHCb Data Flow Run II. Stage 3: stripping

Offline reconstruction is CPU expensive

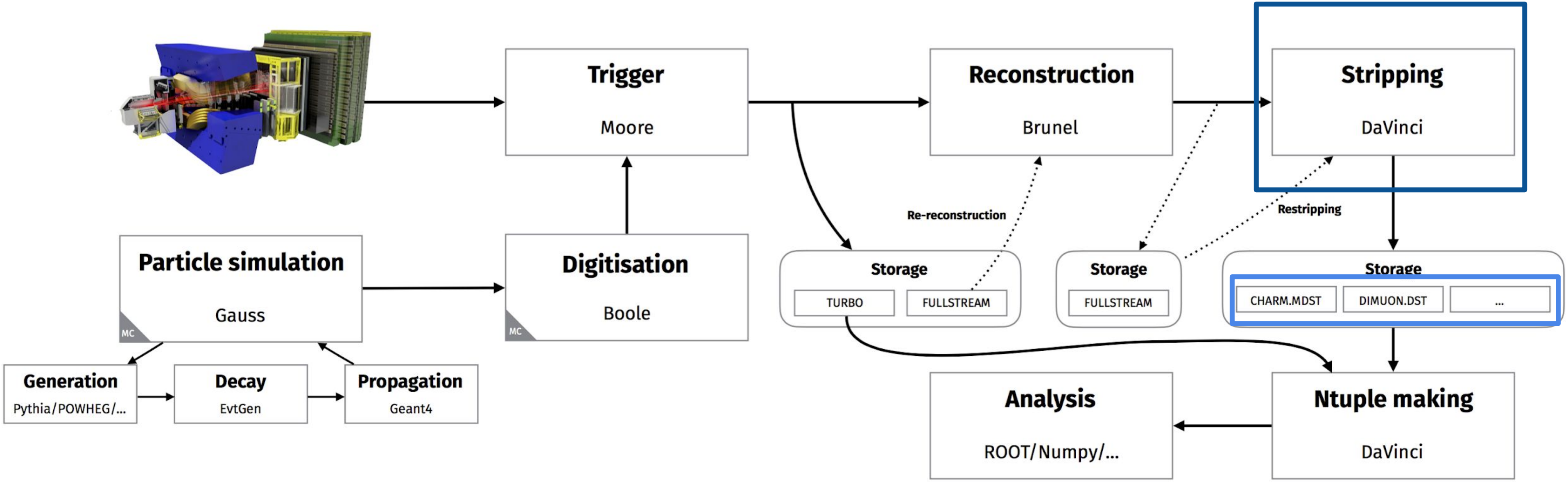
Offline reconstruction produces huge DST files

An analysis user shouldn’t be asked to run the reconstruction her-/himself.

Reconstructed data can be grouped based on the signatures, i.e. can be grouped in the streams of data.

This process we call stripping.

LHCb Data Flow Run II. Stage 3: stripping



Output: DST or mDST

DST = 150 kB/event
mDST = 50 kB/event, only candidate info

[DaVinci on gitlab]

LHCb Data Flow Run II. Stage 3: Stripping

Stripping campaigns are done centrally.

You can find definitions of the stripping lines with cuts etc here

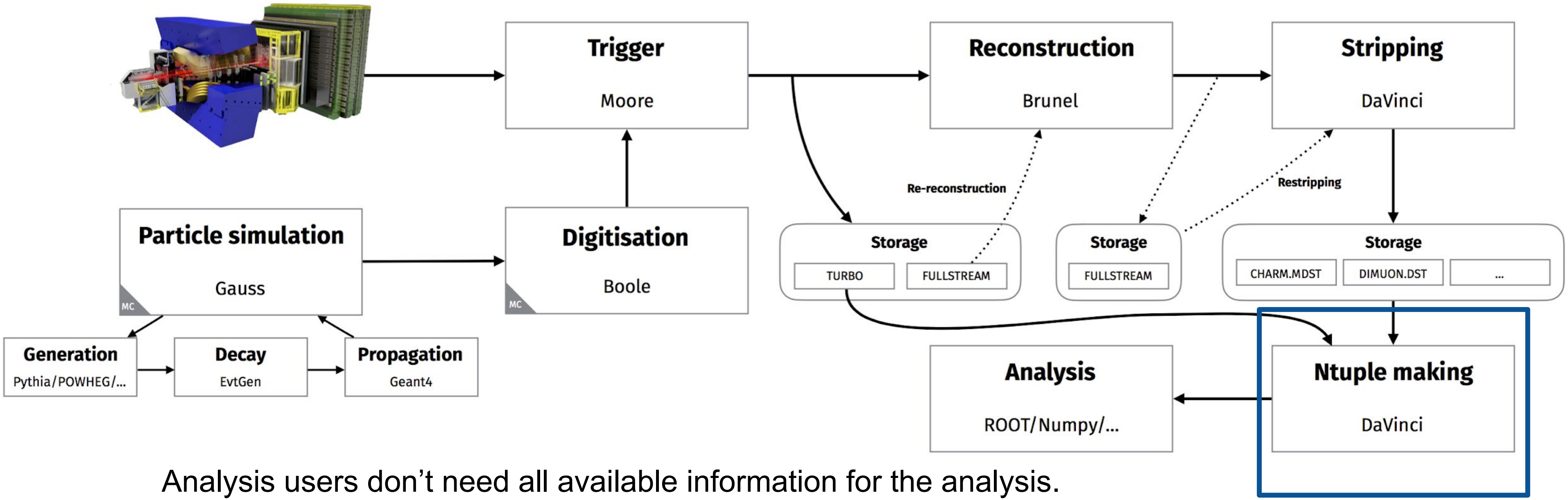
Stripping campaign:

sXrYpZ

- X = restripping campaign (all lines are processed)
- Y = year
- Z = incremental stripping (a few lines are added/fixed and processed)

If you need help with finding the stripping line - ask stripping liasons of your WG

LHCb Data Flow Run II. Stage 4: ntuple making



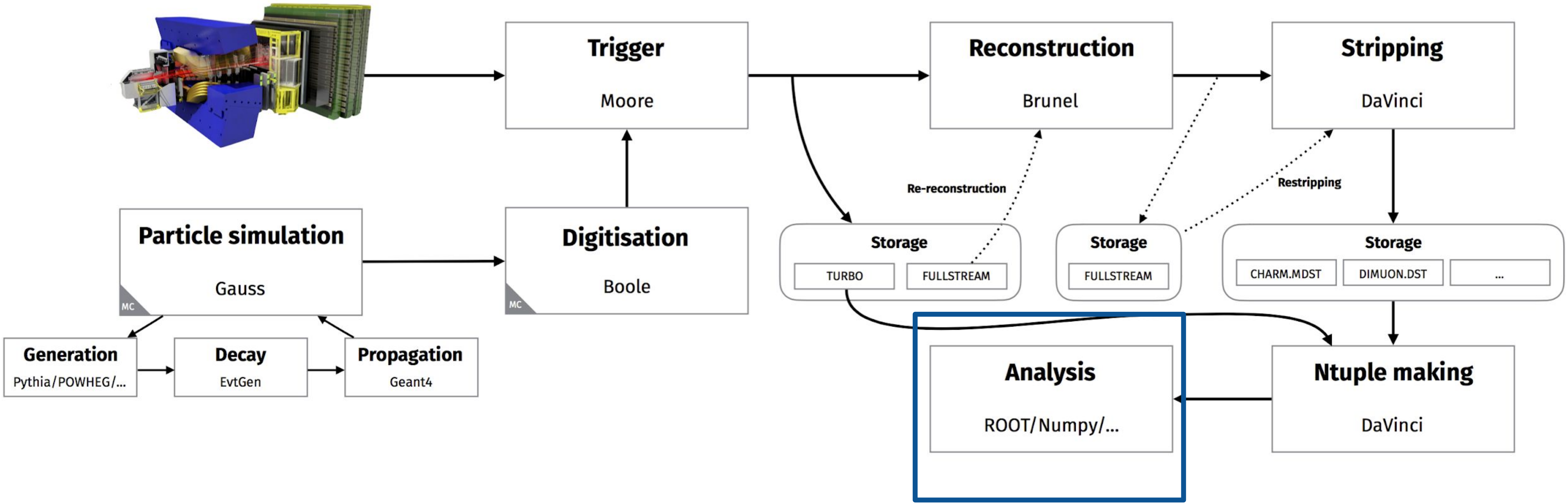
Analysis users don't need all available information for the analysis.

ntuple = ROOT data file

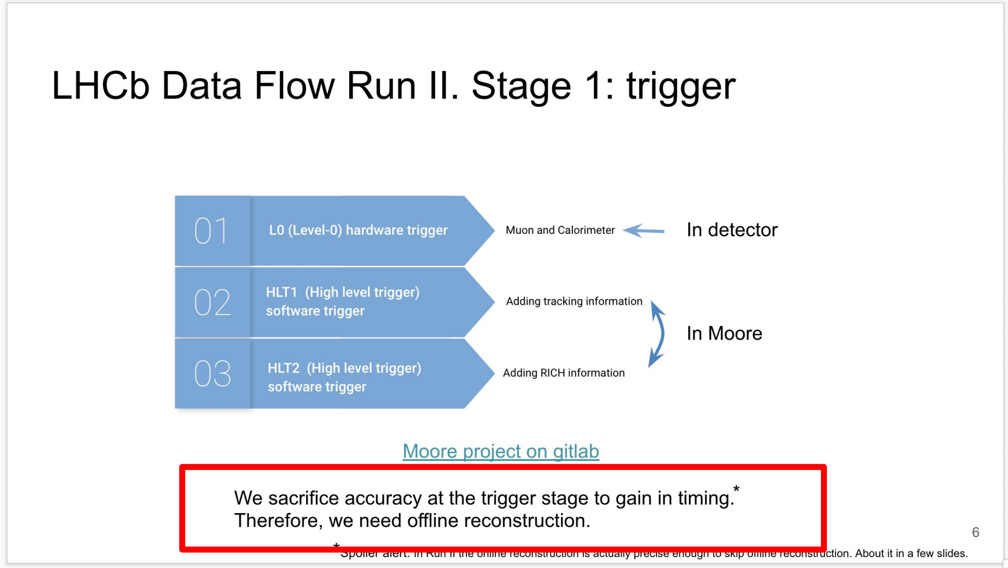
[DaVinci on gitlab]

[Phys on gitlab]

And finally analysis



Run II shortcut: Turbo

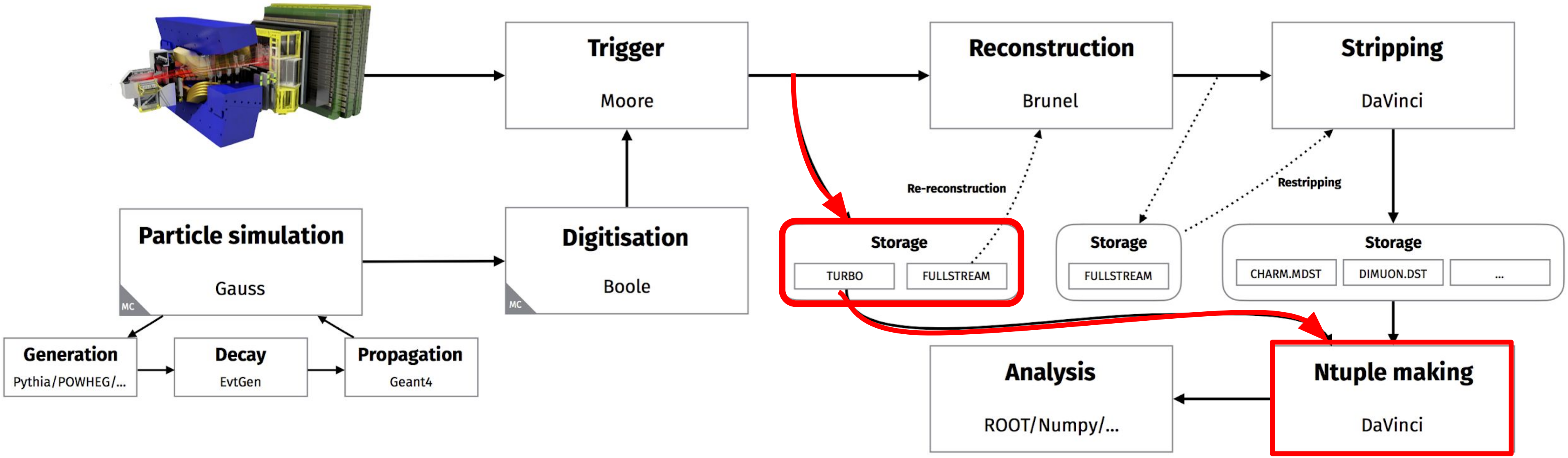


This is **not really true** for Run II

Trigger online reconstruction is **really accurate** in Run II

Why not try to save CPU that is wasted on the offline reconstruction?

Run II shortcut: Turbo



Anything that is not a part of the signal decay is thrown away
No re-reconstruction is possible

Run II shortcut: Turbo

Turbo: save a candidate only

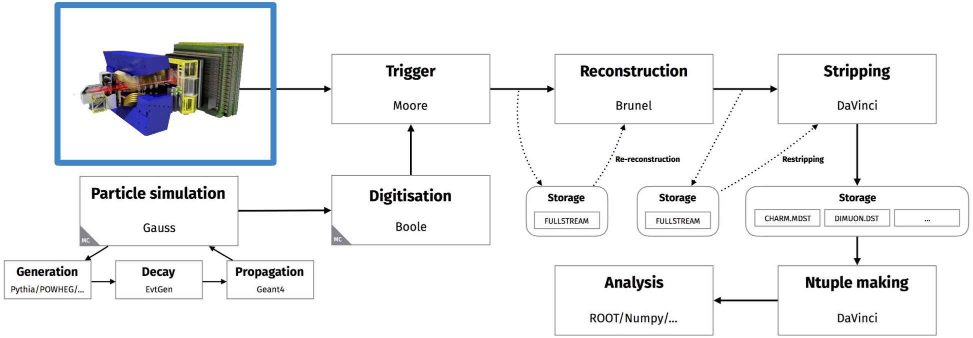
Turbo++: additional track information

TurboSP: you can save an additional information that you want

If you need help with finding the TURBO line - ask trigger liasons of your WG

LHCb Data flow

- ➔ LHC: 40 MHz collision rate ~ 1TB information
- ➔ Our resources are limited



LHCb data flow Run I (2010-2012)

We want to save only the interesting stuff

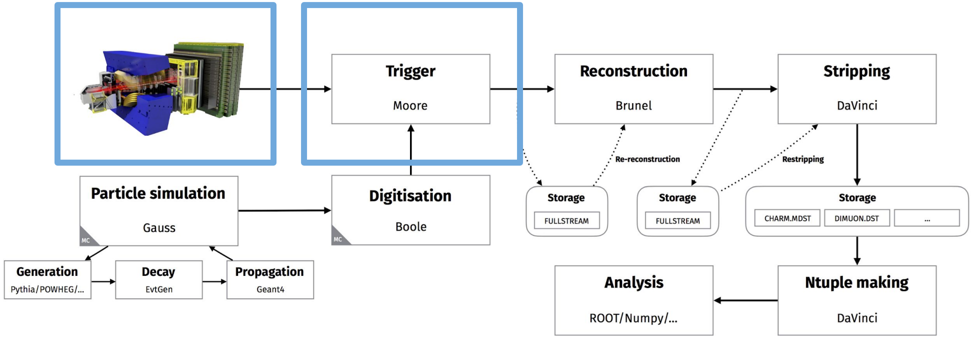
We want to make decision quickly and accurately

We can use limited resources

LHCb Data flow: trigger

- ➔ LHC: 40 MHz collision rate ~ 1TB information
- ➔ We could save only 5 kHz in Run I and 12.5 kHz in Run II
- ➔ This stage is also called “online” reconstruction.

L0 trigger HLT1/2 triggers



LHCb data flow Run I (2010-2012)

3 levels of the LHCb trigger:

- 01 L0 (Level-0) hardware trigger — Muon and Calorimeter ← In detector
- 02 HLT1 (High level trigger) software trigger — Adding tracking information
- 03 HLT2 (High level trigger) software trigger — Adding RICH information

In Moore [Moore project on gitlab]

FULLSTREAM: raw banks of all the subdetectors are saved.

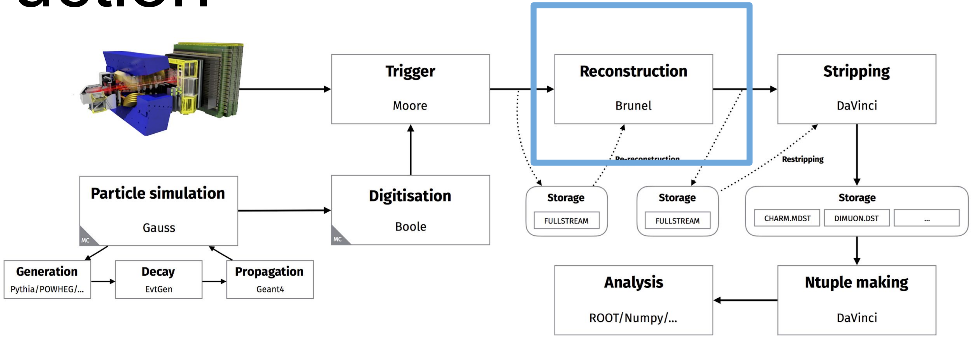
LHCb Data flow: Reconstruction

Because you want to a fast trigger, you have to sacrifice accuracy on the trigger stage.

This is also called “offline” reconstruction.

Next stage is reconstruction:

➔ Here define tracks, clusters, etc…
➔ Brunel on gitlab : reconstruction project
➔ Rec on gitlab : definitions of objects
➔ Output: DST files
➔ CPU expensive

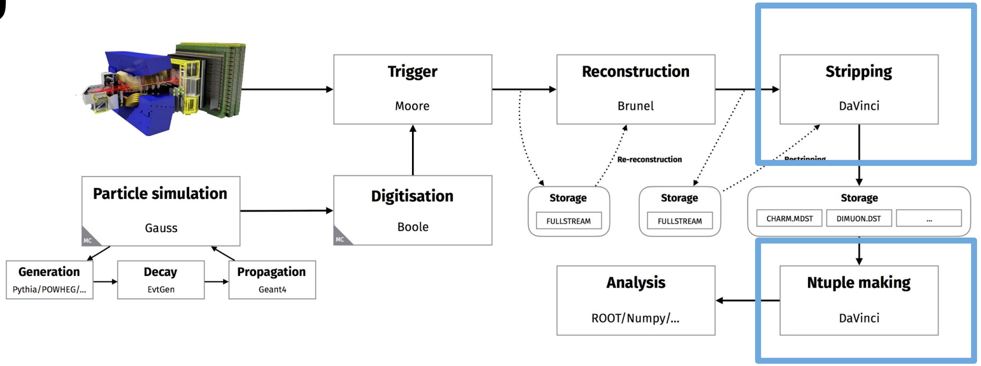


LHCb data flow Run I (2010-2012)

LHCb Data flow: Stripping

Initial DST files are huge. This makes it hard for multiple users to access them when needed. Therefore, data is splitted further in the data streams. Output: DST or mDST files.

- ➔ DaVinci on gitlab : stripping and ntuple making
- ➔ Stripping campaigns are done centrally



LHCb data flow Run I (2010-2012)

> You can find definitions of the stripping line with cuts etc here

Stripping campaign:

sXrYpZ

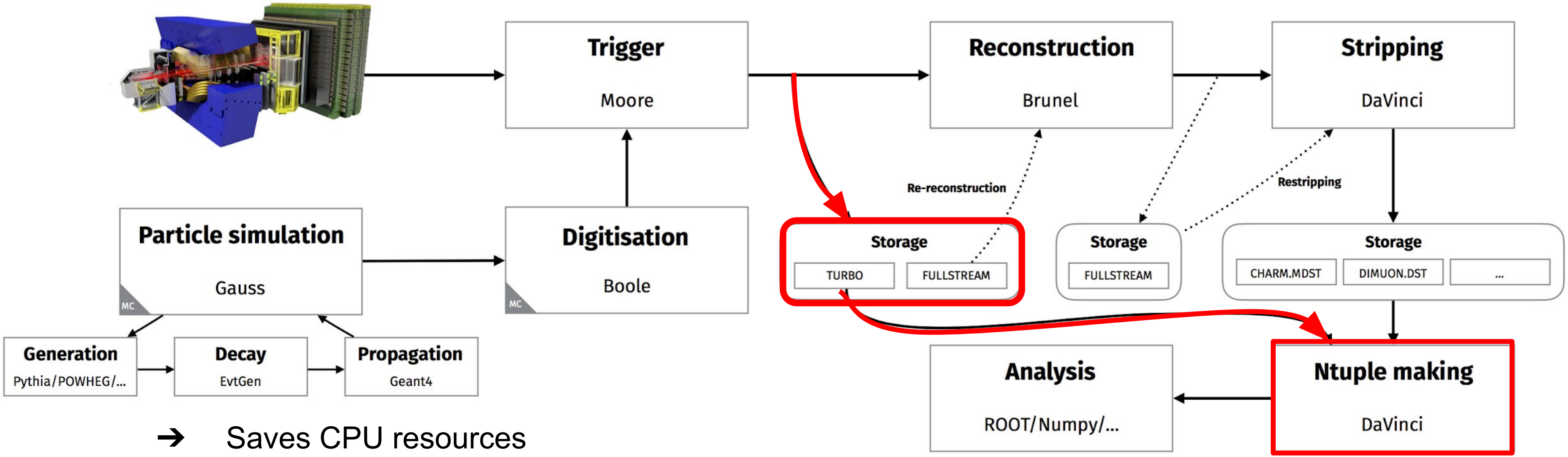
- X = restripping campaign (all lines are processed)
- Y = year
- Z = incremental stripping (a few lines are added/fixed and processed)

If you need help with finding the stripping line - ask stripping liasons of your WG

ntuple = ROOT data file

DST = 150 kB/event
mDST = 50 kB/event, candidate info

LHCb Data flow: Turbo



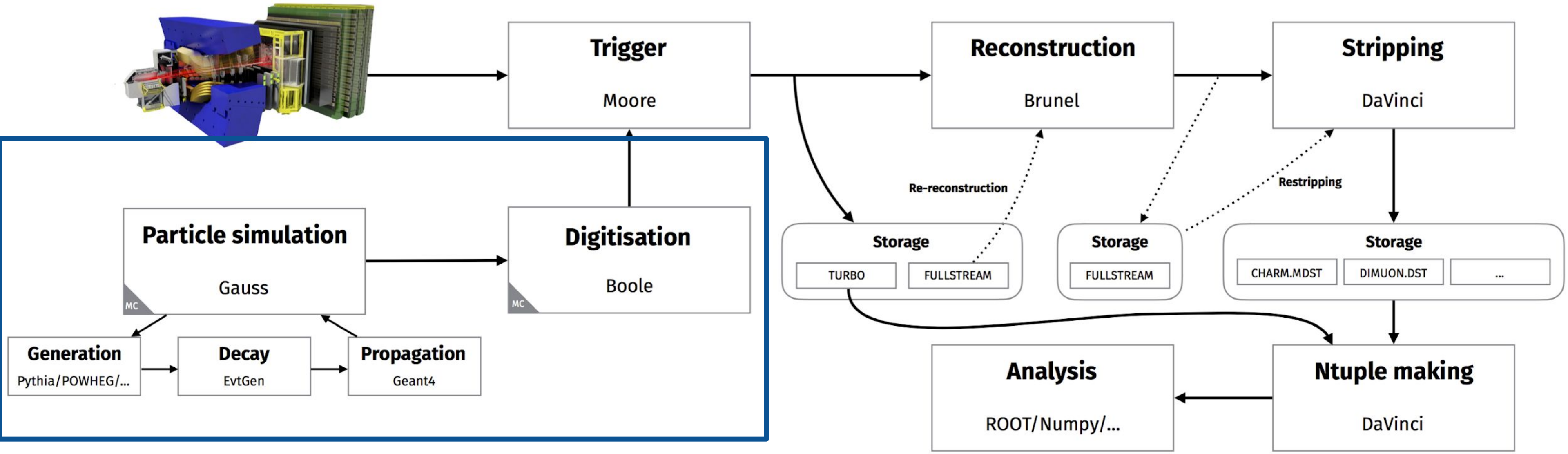
- Saves CPU resources
- You don't need Brunel reconstruction any more! HLT2 is accurate enough!
- Anything that is not a part of decay is thrown away
- No re-reconstruction is possible!

If you need help with finding the TURBO line - ask trigger liasons of your WG

Turbo: save a candidate only
Turbo++: additional track information
TurboSP: you can save an additional information that you want.

Simulation 

Simulation



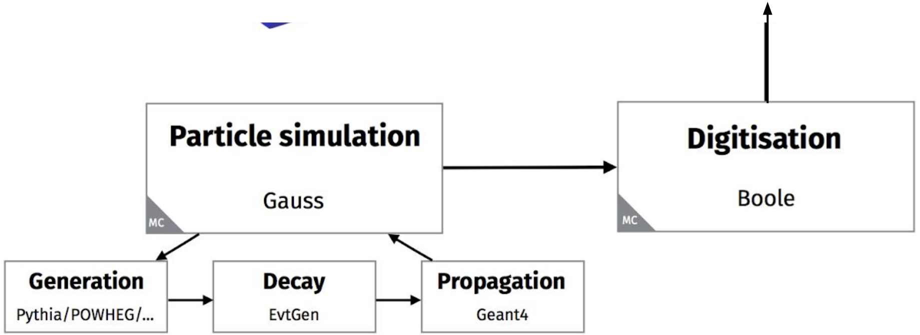
Simulated Monte Carlo events pass the same reconstruction sequence as data.

Simulation

Simulation is controlled by the **Gauss**.

- ➔ Hard process generation:
  - ◆ Pythia8
  - ◆ SuperChic
  - ◆ BcVegPy
  - ◆ ...
- ➔ Decay: EvtGen
- ➔ Detector response: Geant4
- ➔ Detector response digitalization: Boole

Software Trigger (Moore)



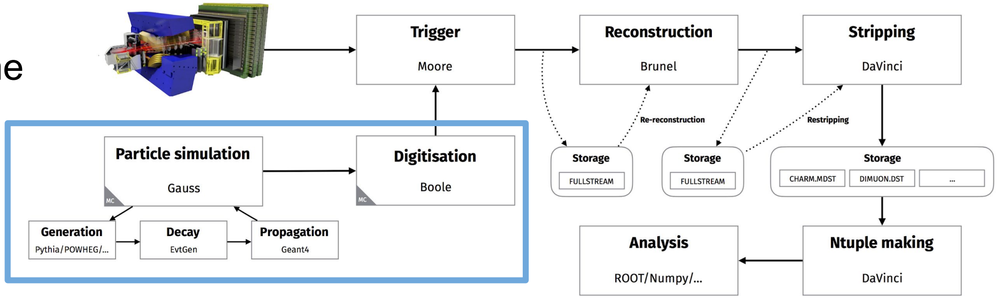
A bit more on simulations in the second-analysis-steps

The process option file is called dec file.
Examples of dec files can be found here

If you need help with Monte Carlo - ask simulation liasons of your WG

LHCb Data flow: Monte Carlo

Simulated Monte Carlo events pass the same reconstruction sequence as data.



LHCb data flow Run I (2010-2012)

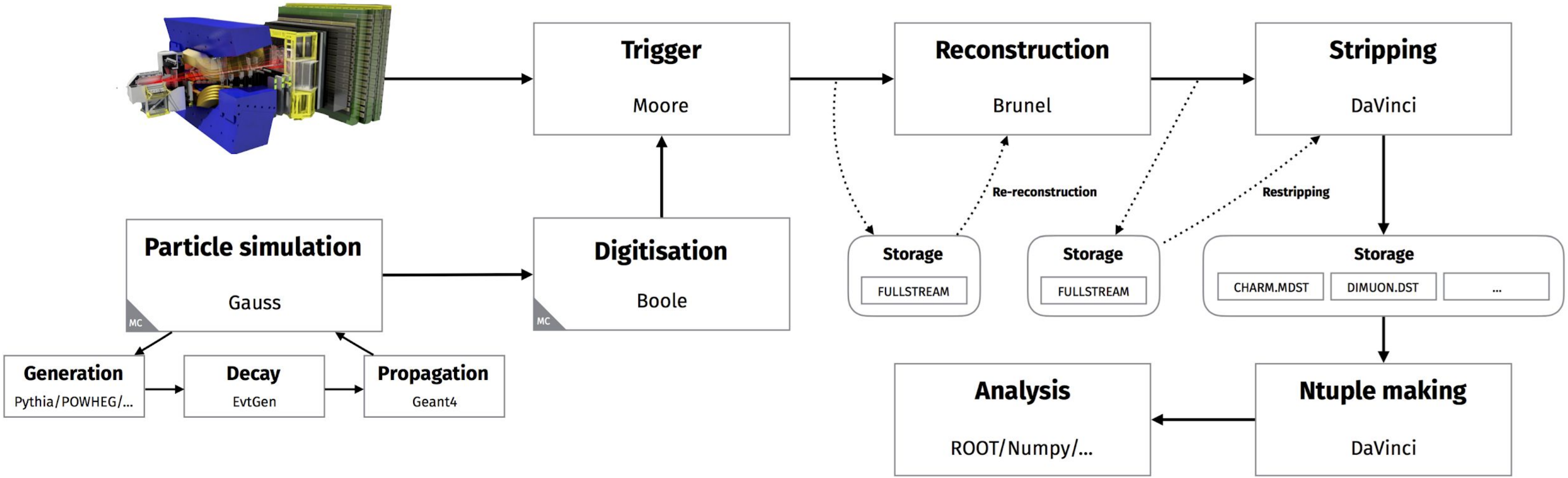
1. Gauss: controls simulation, calls generators like Pythia8 (SuperChic, BcVegPy, GenXicc…), EvtGen and Geant4
2. Boole: digitalization to match the detector signal

The process option file is called dec file. Examples of dec files can be found here

A bit more on simulations in the second-analysis-steps

What was done in Run I?📜
2010-2012

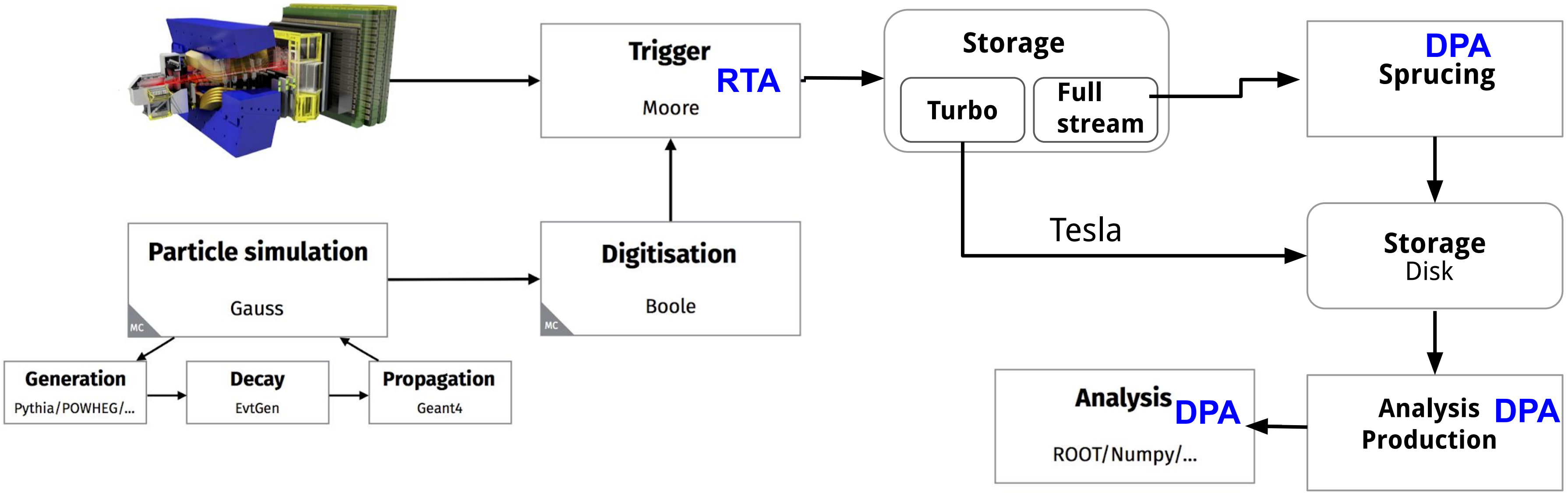
LHCb Data Flow Run I



There was no Turbo “shortcut”.
The online reconstruction resolution was worse, that in Run II.
Therefore, offline reconstruction was a “must”.

What will be done in Run III? 🔮
2021-2024

LHCb Data Flow Run III (as planned)



Turbo/Turbo SP are default in Run III

Intro to LHCb software 👦💻

Gaudi : 5 important concepts

LHCb software is based on the Gaudi framework

Gaudi Manual
Gaudi doxygen

1. **Event loop :** Gaudi allows to process events one by one. Setup by `gaudirun.py`
2. **Transient Event Store:** location of different objects in Gaudi. For example, best tracks can be found in the default location: `/Event/Rec/Track/Best`
3. **Algorithm:** C++ class that allows to perform certain action with an event. Example, PVResMonitor
4. **Tools:** functions that are shared between the Algorithms. Example, MeasurementProvider
5. **Options:** configuration of Tools and Algorithms, as well as their order, in a python option file. Example, HLT2 sequence example

DaVinci👨‍🎨