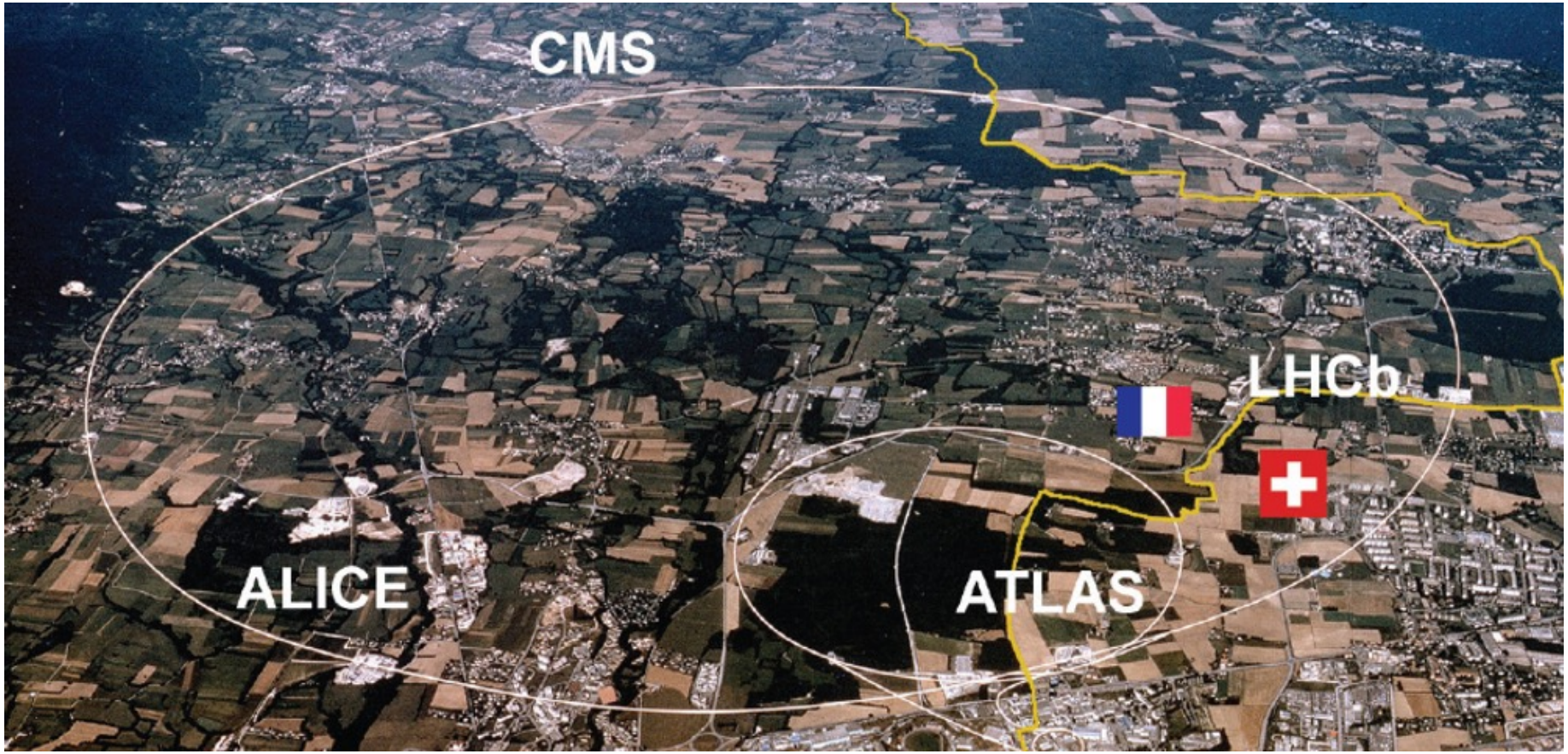
Offline analysis software for LHCb's Run III

Abhijit Mathad, CERN
On behalf of LHCb
CHEP 2024, Krakow, Poland

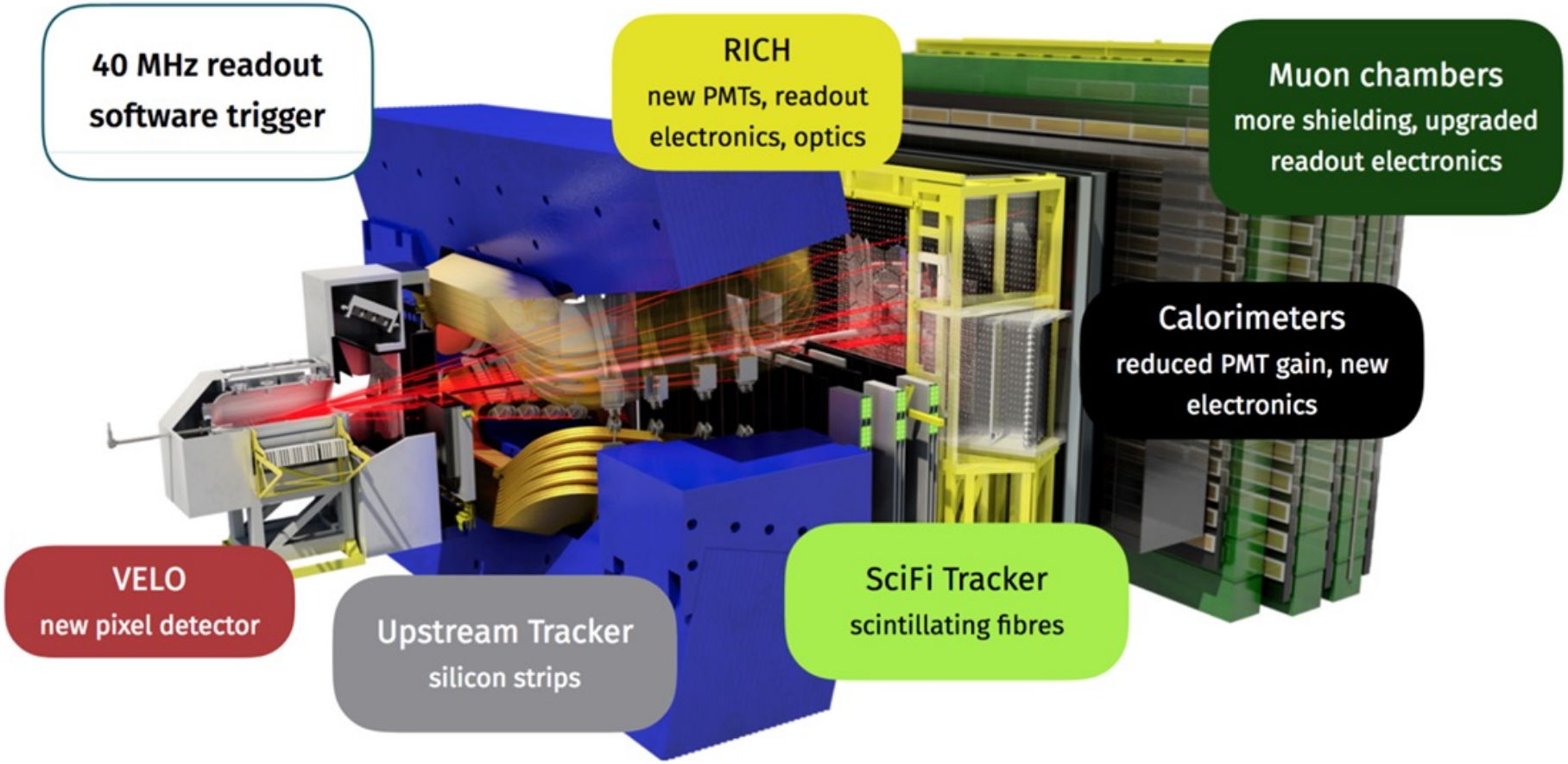
LHCb detector is located in France and specialises in studies of beauty and charm hadrons.



It has undergone a major upgrade ahead of LHC Run III (2022-2026)!

Brand new detector

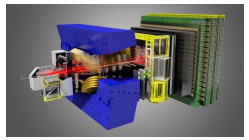
Instantaneous proton-proton luminosity: $L_{inst}^{Run\ III} = 5 \times L_{inst}^{Run\ II}$



Brand new software

Run 2 (Simplified)

After Hadrware trigger



0.1 TB/s ↓ 1 MHz

Trigger (Moore)

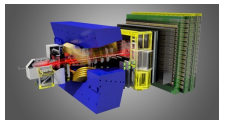
Turbo

Offline Reconstruction (Brunel)

0.6 GB/s

Offline (DaVinci)

Run 3



5 TB/s ↓ 40 MHz

Fully software trigger (Allen and Moore)

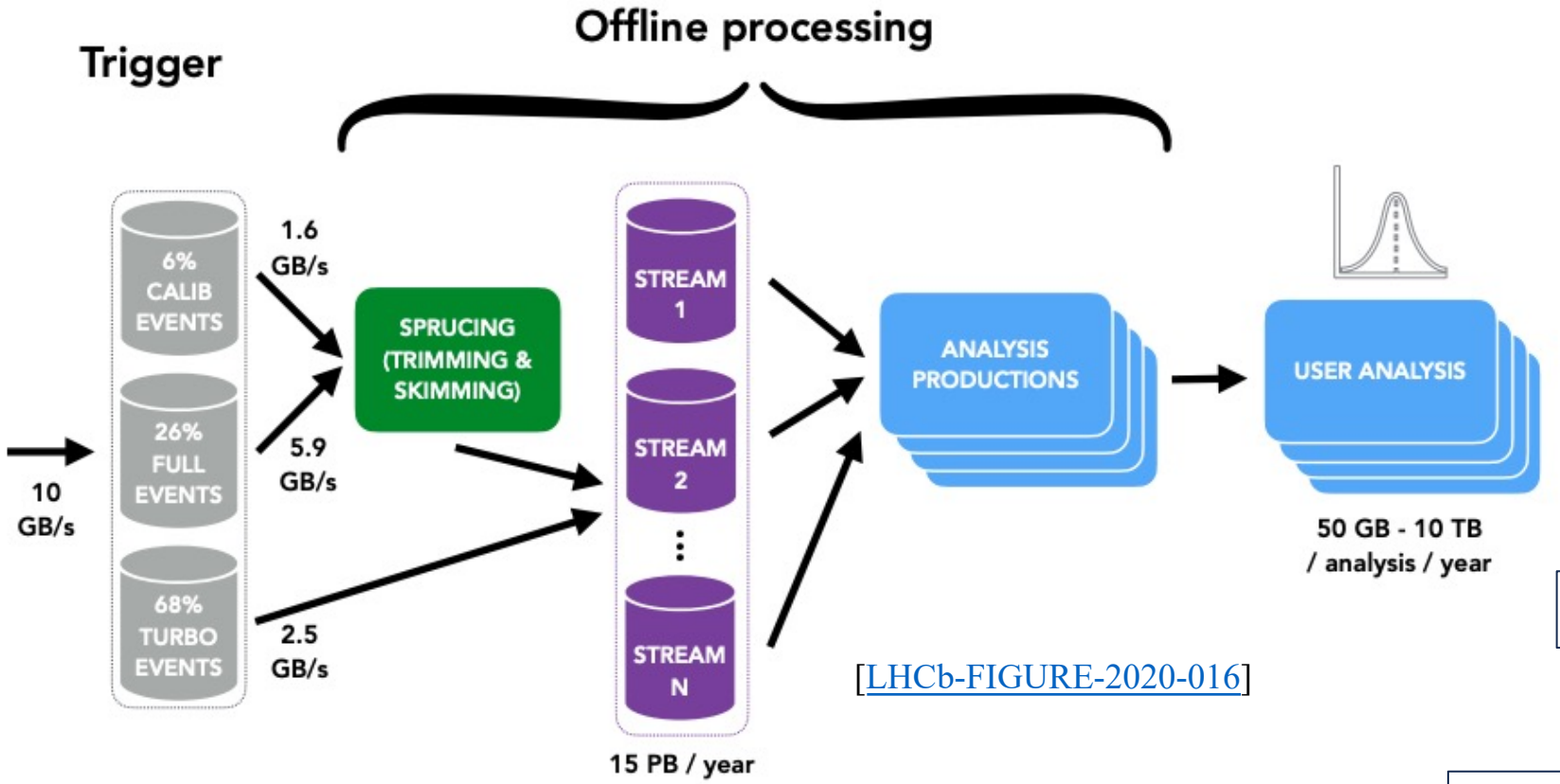
10 GB/s

Offline (Moore, DaVinci)

* Frontend software projects

Major overhaul of trigger (online) and **offline** software!

Facets of (Offline) Data Processing



[LHCb-FIGURE-2020-016]

See trigger talk by Alessandro Scarabotto.

WP1: Sprucing

→ See talk by Nicole Skidmore

WP2: Analysis Production

→ See talk by Nicole Skidmore

WP3: Offline analysis tools

→ This talk

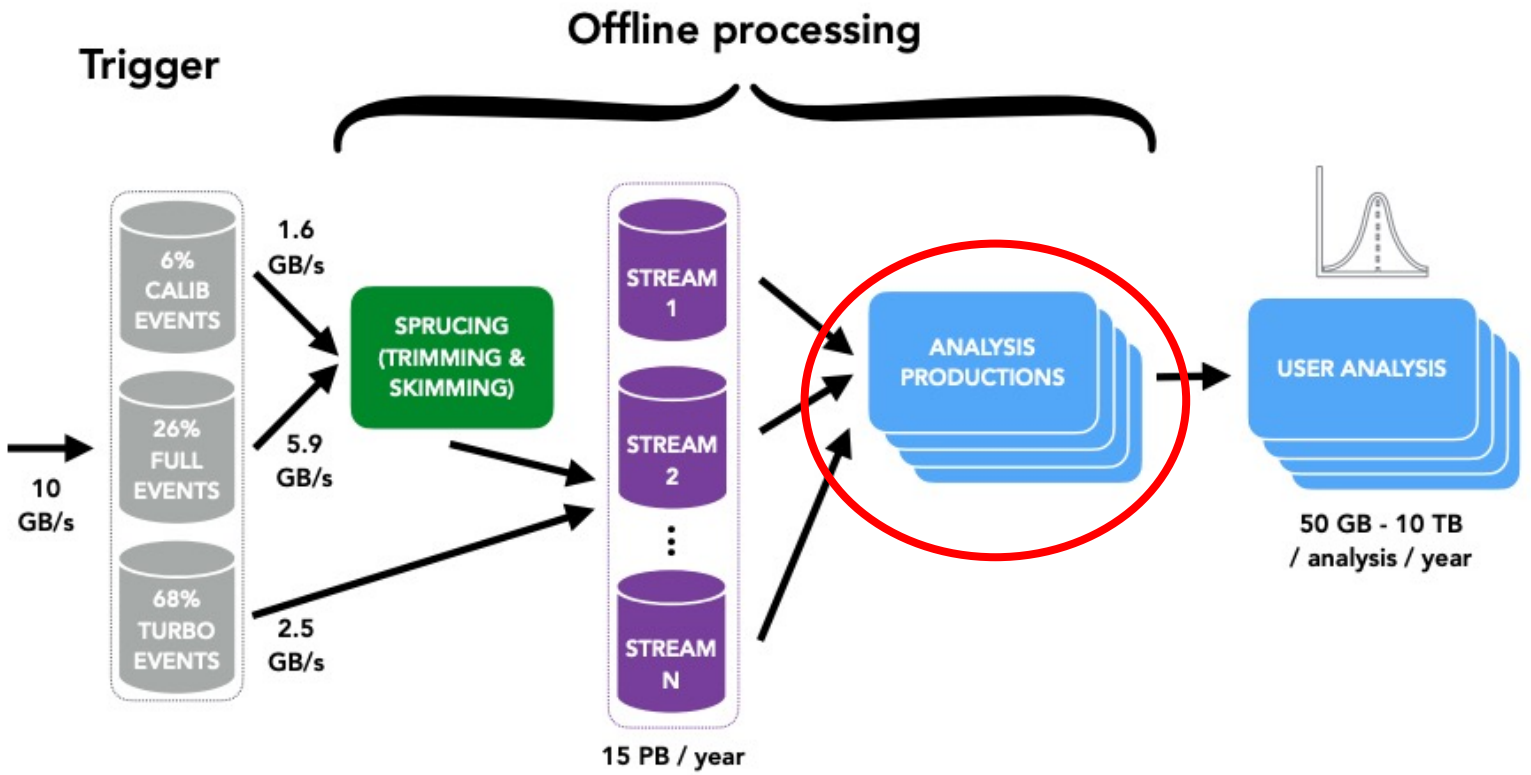
WP4: Legacy software and data

→ See talk by Nathan Grieser

WP5: Analysis preservation and open data

→ See talk by Piet Nogga and talk by Mindaugas Sarpis

Facets of (Offline) Data Processing



WP3: Offline analysis tools

→ Software used **DaVinci**.

Git repository link

DaVinci goal: Read and process data/simulation post trigger to generate flat ROOT files for analysis, accessible via central production!

Umbrella of DaVinci

Components

FunTuple

Bring together various tools to write ROOT ntuples.

Framework

Defining the input/output and configuring a job

Unit tests

Controlled tests with data/simulation to validate the processing chain.

Support

Documentation, Examples, Tutorials and day-to-day support.

Algorithms

Decay Tree Fit

Fit the entire decay chain to improve resolution

TisTos

Triggered on signal or indpendent of signal?

MCTruth

What is the truth information?

Flavour Tagging

What is the flavour of neutral B?

… etc

FunTuple

Comput Softw Big Sci 8, 6 (2024)

Computing and Software for Big Science (2024) 8:6
https://doi.org/10.1007/s41781-024-00116-1

BRIEF REPORT

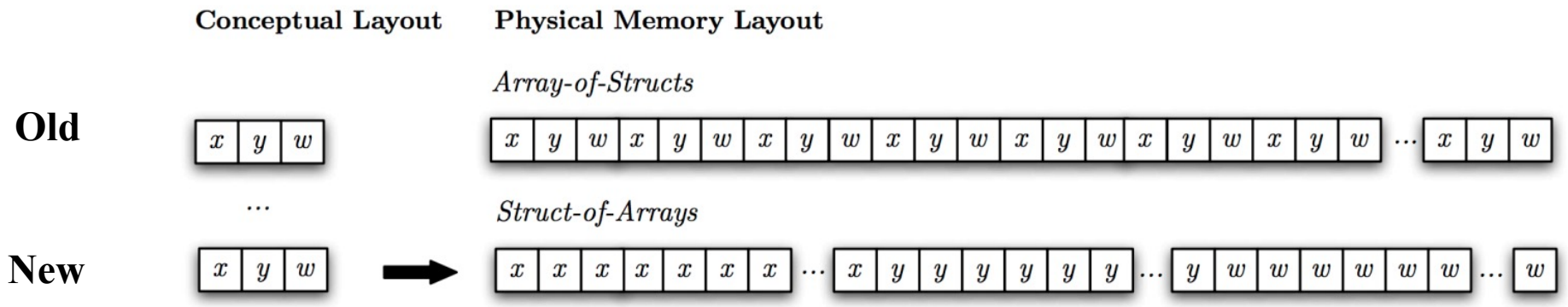
Check for updates

FunTuple: A New N-tuple Component for Offline Data Processing at the LHCb Experiment

Abhijit Mathad[1,2] · Martina Ferrillo[1] · Sacha Barré[2,3] · Patrick Koppenburg[4] · Patrick Owen[1] · Gerhard Raven[4,5] · Eduardo Rodrigues[6] · Nicola Serra[1]

- Agnostic to the event model
- Equivalence between online and offline
- Simple interface with other algorithms
- Complete user flexibility

Online software **slowly** moving towards Simd data stuctures for fast processing.



[Talk by Sevda]

FunTuple is a templated C++ algorithm with a Python frontend, agnostic to any simulation or event model.

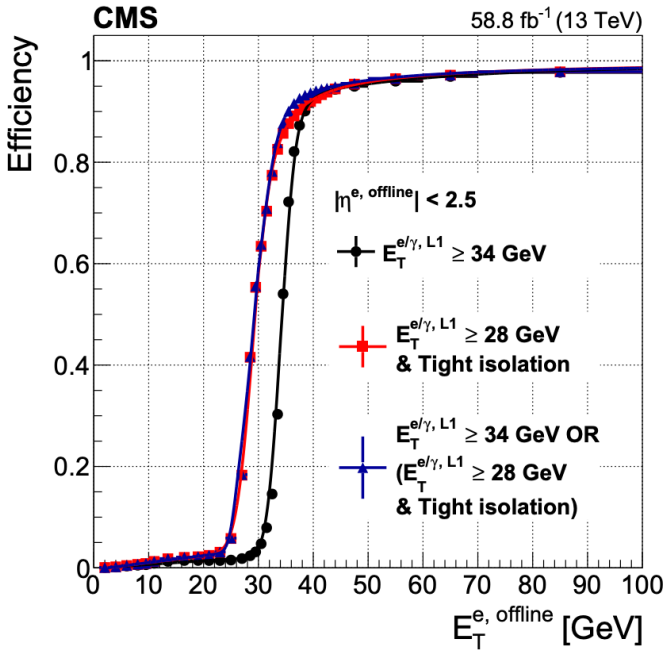
Agnostic to the event model

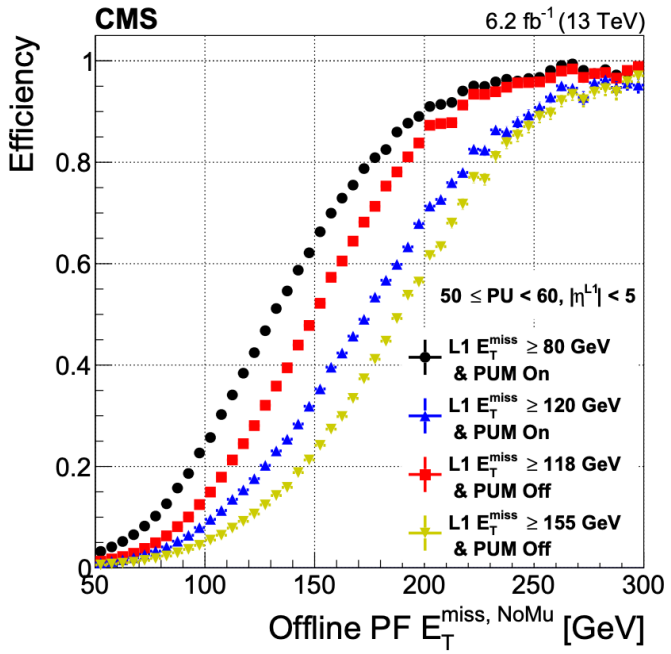
Equivalence between online and offline

Simple interface with other algorithms

Complete user flexibility

Online
≡
Offline





[V Gligrov and V Rekovic]

Online
≠
Offline

Careful efficiency modelling required

FunTuple employs the same Throughput Oriented (ThOr) functors as the trigger

Agnostic to the event model

Equivalence between online and offline

Complete user flexibility

Simple interface with other algorithms

- **Over 400 ThOr functors have been developed for both online and offline.**

- **Many more can be created using composition (via @ operator).**

```
PVX = F.X_COORDINATE @ F.POSITION @ F.MC_PRIMARYVERTEX
```

- **Supports basic (*, /, +, -) and advanced (fmath) operations.**

```
PT = fmath.sqrt(fmath.pow(PX, 2) + fmath.pow(PY,2))
out= fmath.where(cond, func_A, func_B)
```

- **Functors are JIT compiled optimising them at runtime.**

Agnostic to the event model

Equivalence between online and offline

Complete user flexibility

Simple interface with other algorithms

Run 2 Limitations:

- Data bundled with limited flexibility in writing to nTuples **at run-time**.
- Analysis nTuples reached 3 TB in size

Run 3 and Beyond:

- Need a scalable solution for increasing luminosity and data volume.

Agnostic to the event model

Equivalence between online and offline

Complete user flexibility

Simple interface with other algorithms

Answer: Customisable FunctorCollections

Run 2 Limitations:

- Data bundled with limited flexibility in writing to nTuples **at run-time**.
- Analysis nTuples reached 3 TB in size

Run 3 and Beyond:

- Need a scalable solution for increasing luminosity and data volume.

```
"EventInfo",
"SelectionInfo",
"HltTisTos",
"Kinematics",
"MCHierarchy",
"MCKinematics",
"MCVertexInfo",
"MCPromptDecay",
"MCReconstructible",
"MCReconstructed",
"ParticleIsolation",
"ConeIsolation",
"VertexIsolation",
"NeutralCaloInfo",
"ChargedCaloInfo",
"DecayTreeFitterResults",
"ParticleID",
"MCPrimaries",
"RecSummary",
"FlavourTaggingResults",
```

Example storing trigger information

```python
def SelectionInfo(*, selection_type: HltSourceID,
                  trigger_lines: list[str]) -> FunctorCollection:
    # Ensure selection_type is of type HltSourceID
    selection_type = HltSourceID(selection_type)

    # Get decision reports
    dec_report = get_decreports(selection_type)

    # Ensure each trigger line ends with "Decision"
    trigger_lines = [s if s.endswith("Decision") else s + "Decision"
                     for s in trigger_lines]

    # Create FunctorCollection and add TCK and decisions
    trigger_info = FunctorCollection()
    trigger_info[f'{selection_type.name}_TCK'] = F.TCK(dec_report)
    trigger_info.update({trigger_line: F.DECISION(dec_report, trigger_line)
                         for trigger_line in trigger_lines})
    return trigger_info
```

Agnostic to the event model

Equivalence between online and offline

Complete user flexibility

Simple interface with other algorithms

FunTuple has simple interface with all algorithms, for example let's consider truth matching.

Truth matching of reconstructed objects is crucial: resolution, acceptance, etc.

Agnostic to the event model

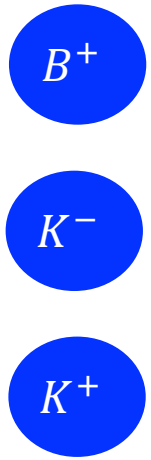
Equivalence between online and offline

Complete user flexibility

Simple interface with other algorithms

ThOr functor for transverse momentum (reco)

$\{\text{“}RECO\_PT\text{”} : F.PT\}$



Reconstructed particles

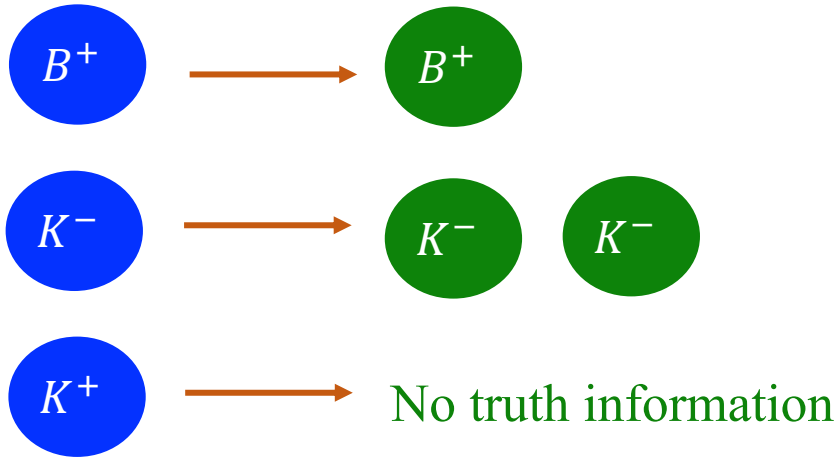
Agnostic to the event model

Equivalence between online and offline

Complete user flexibility

Simple interface with other algorithms

Build one-to-many relations



Reconstructed particles

Truth particles

ThOr functor for transverse momentum (reco)

$$\{\text{“}RECO\_PT\text{”} : F.PT\}$$

ThOr functor for transverse momentum (truth)

$$MCTRUTH = MCTruthAlg(reco\_parts, mc\_parts)$$

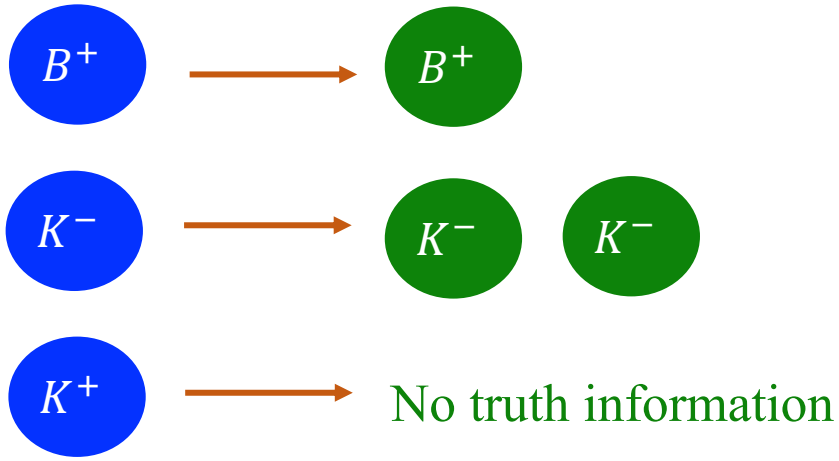
Agnostic to the event model

Equivalence between online and offline

Complete user flexibility

Simple interface with other algorithms

Build one-to-many relations



Reconstructed particles

Truth particles

ThOr functor for transverse momentum (reco)

{"RECO_PT" : F.PT}

ThOr functor for transverse momentum (truth)

MCTRUTH = MCTruthAlg(reco_parts, mc_parts)

{"TRUE_PT": MCTRUTH(F.PT)}

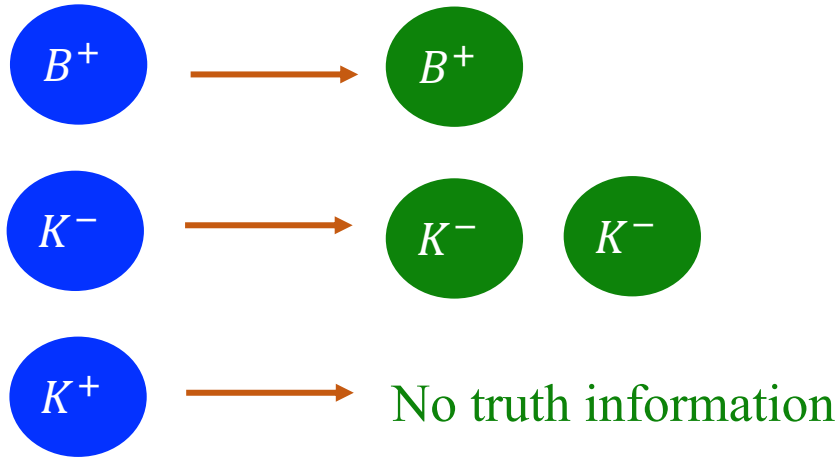
Agnostic to the event model

Equivalence between online and offline

Complete user flexibility

Simple interface with other algorithms

Build one-to-many relations



Reconstructed particles

Truth particles

Can store any custom data stucture from functors (Arrays, Matrices, Maps, etc).

Cases where relations non-existent, an internal fail-safe inplace for all supported types.

Agnostic to the event model

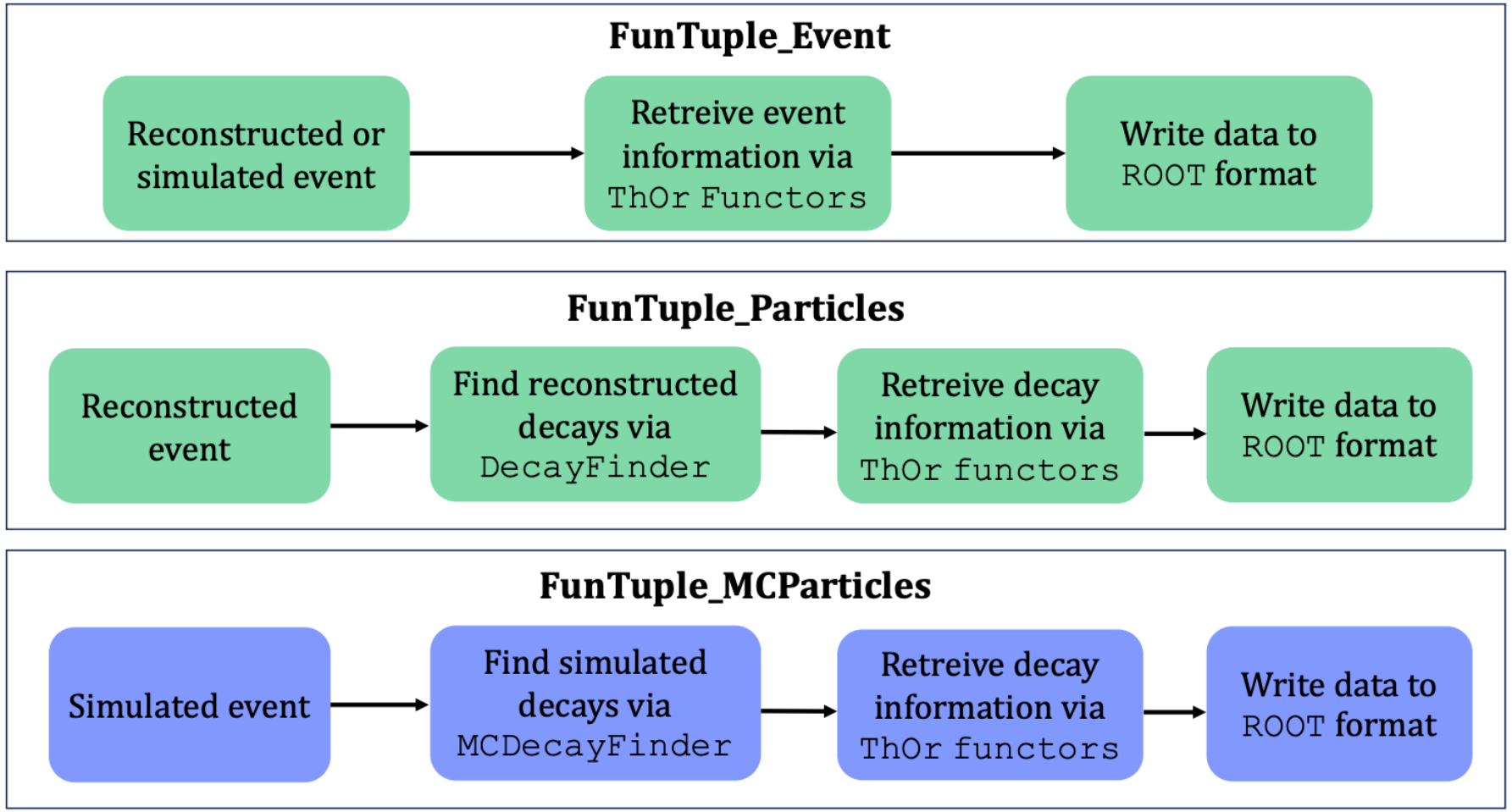
Equivalence between online and offline

Complete user flexibility

Simple interface with other algorithms

FunTuple: Data flow

Comput Softw Big Sci 8, 6 (2024)



New improved decay finder also developed [CERN-STUDENTS-NOTE-2022-211]

DaVinci configuration

- **Configuration modernised**: Based on PyConf (algorithm scheduler) modules, based on click and compatible with python 3.

```
./run lbexec function options
```

```
function
        Function to call with the options that will return the
configuration. Given in the form 'my_module:function_name'.

options
        YAML data to populate the Application.Options object
with. Multiple files can merged using 'file1.yaml+file2.yaml'.
```

Unit and “Physics” tests

Tests check configurations, output, nTuple values, FunctorCollections behaviour, and FunTuple with different event models.

40 unit tests

pytest

```python
@pytest.mark.parametrize(
    "func_name,name_argument,expected_type", list_function_tuple_name
)
def test_variables_all_type_FunctorCollection(func_name, name_argument, expected_type):
    """
    Check that the 'variables' (values) are all of type FunctorCollection.
    """
    fields = dict(B="A -> ^B C", C="A -> B ^C")
    variables = {
        "B": FunctorCollection({"THOR_FourMom_P": F.FOURMOMENTUM}),
        "C": {"THOR_FourMom_P": F.FOURMOMENTUM},
    }

    with pytest.raises(TypeError):
        _ = getattr_wrapper(
            this_module,
            func_name,
            name=name_argument + "_{hash}",
            tuple_name=name_argument,
            fields=fields,
            variables=variables,
            inputs=data_handle(expected_type),
        )
```

100 “physics” tests

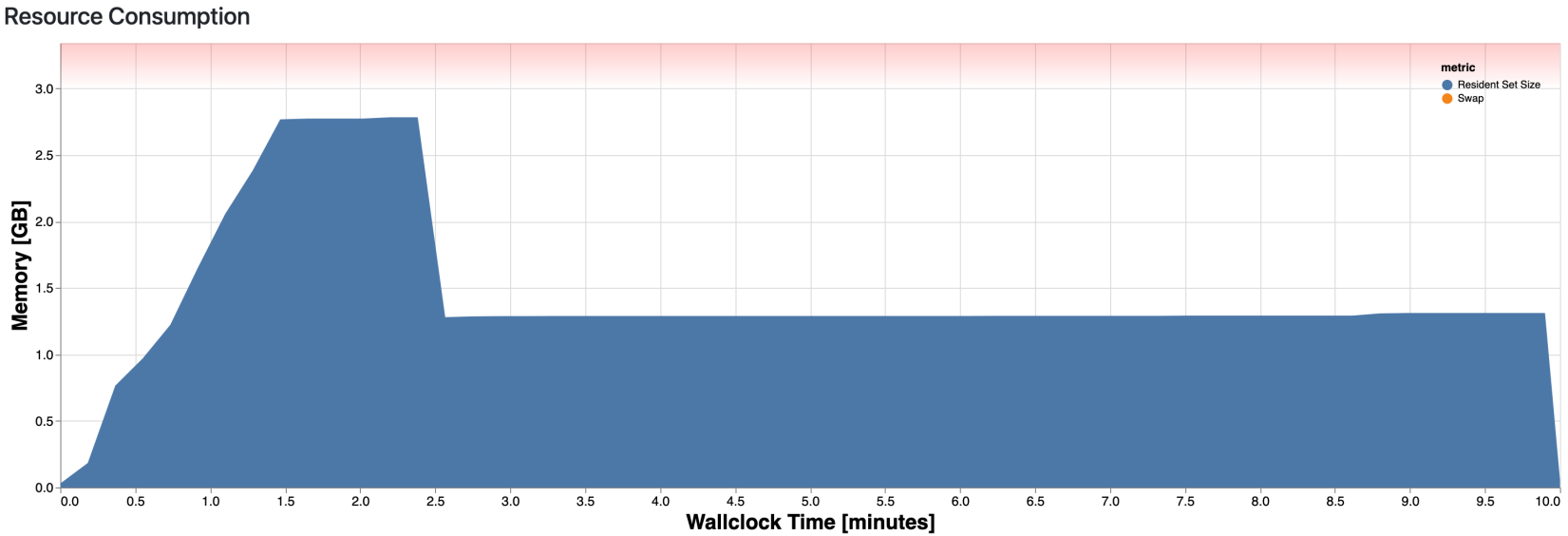
Gaudi QMT

```
<!DOCTYPE extension  PUBLIC '-//QM/2.3/Extension//EN'
  'http://www.codesourcery.com/qm/dtds/2.3/-//qm/2.3/extension//en.dtd'>
<extension class="GaudiTest.GaudiExeTest" kind="test">
  <argument name="program"><text>lbexec</text></argument>
  <argument name="args"><set>
    <text>DaVinciExamples.tupling.option_davinci_tupling_eventinfo:main</text>
  </set></argument>
  <argument name="options_yaml_fn"><text>$DAVINCIEXAMPLESROOT/example_data/Upgrade_LbToLcmunu.yaml</text></argument>
  <argument name="extra_options_yaml"><text>
    ntuple_file: tuple_LbToLcmunu.root
    print_freq: 1
  </text></argument>
  <argument name="reference"><text>../refs/test_davinci_tupling_eventinfo.ref</text></argument>
  <argument name="error_reference"><text>../refs/empty.ref</text></argument>
  <argument name="validator"><text>
from DaVinciTests.QMTest.DaVinciExclusions import preprocessor, counter_preprocessor, remove_known_warnings
from PyConf.components import findRootObjByDir
validateWithReference(preproc = preprocessor, counter_preproc = counter_preprocessor)

from pathlib import Path
from ROOT import TFile

B_vars_stored = ['nLongTracks', 'nPVs', 'EVENTNUMBER', 'RUNNUMBER', 'Lb_PT', 'GPSTIME', 'BUNCHCROSSING_TYPE', 'ODINTCK' ]

#sort the expected vars
B_vars_stored = sorted(B_vars_stored)

#open the TFile and TTree
ntuple = Path('./tuple_LbToLcmunu.root')
if not ntuple.is_file(): raise Exception(f"File: {ntuple} does not exist!")
f      = TFile.Open(ntuple.name)
t_B    = findRootObjByDir(f, 'Tuple', 'DecayTree')

#sort the stores vars
b_names = sorted([b.GetName() for b in t_B.GetListOfLeaves()])

B_excluded_1 = set(B_vars_stored) - set(b_names)
B_excluded_2 = set(b_names) - set(B_vars_stored)
if len(B_excluded_1) != 0:
  raise Exception('Number of stored variables is less than what is expected. The extra variables expected are: ' , B_excluded_1)
if len(B_excluded_2) != 0:
  raise Exception('Number of stored variables is greater than what is expected. The extra variables stored are: ', B_excluded_2)

f.Close()
print('Test successfully completed!')
ntuple.unlink()
countErrorLines({"FATAL": 0, "WARNING": 0, "ERROR": 0},
                stdout=remove_known_warnings(stdout))

  </text></argument>
</extension>      You, 1 second ago • Uncommitted changes
```

Test coverage for both FunTuple and related tools stands at an impressive 100%!

Performance

- Recording 740 observables for 1000 events takes 3 minutes.
- JIT compilation of 200 functors takes 84 seconds, with high memory usage.
- Functor caching reduces offline processing overhead, but optimizing its use in distributed computing requires further work.



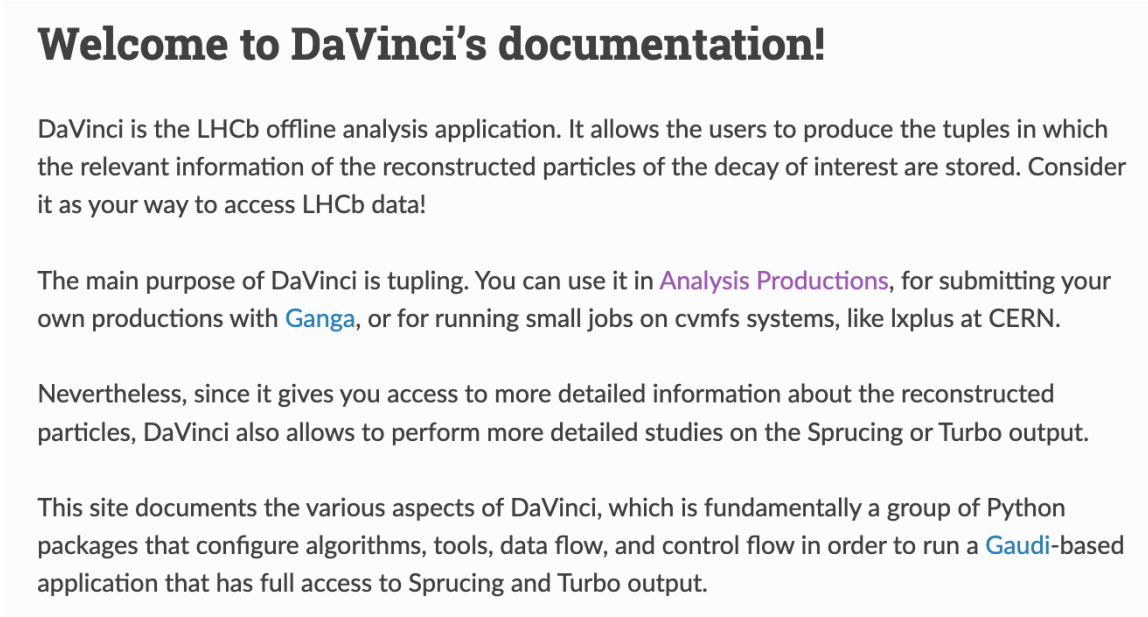
	Maximum	Average	Largest Increase [per 30s]
Resident Set Size	2.78 GB	1.41 GB	582 MB at 0 min
Swap	—	—	— at 0 min

[Taken from Analysis Productions]

Documentation and Support

Contains DaVinci Examples and Tutorials

[DaVinci Documentation via Sphinx]



Welcome to DaVinci's documentation!

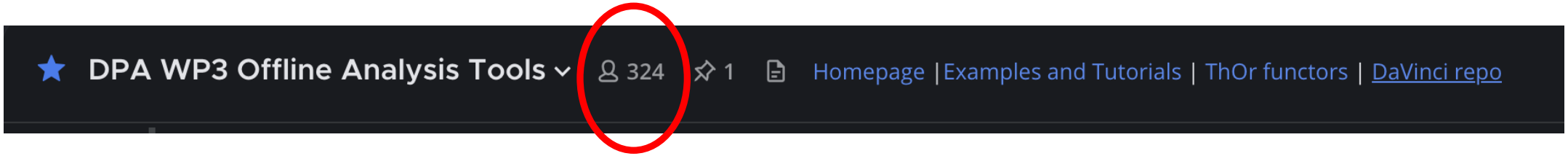
DaVinci is the LHCb offline analysis application. It allows the users to produce the tuples in which the relevant information of the reconstructed particles of the decay of interest are stored. Consider it as your way to access LHCb data!

The main purpose of DaVinci is tupling. You can use it in Analysis Productions, for submitting your own productions with Ganga, or for running small jobs on cvmfs systems, like lxplus at CERN.

Nevertheless, since it gives you access to more detailed information about the reconstructed particles, DaVinci also allows to perform more detailed studies on the Sprucing or Turbo output.

This site documents the various aspects of DaVinci, which is fundamentally a group of Python packages that configure algorithms, tools, data flow, and control flow in order to run a Gaudi-based application that has full access to Sprucing and Turbo output.

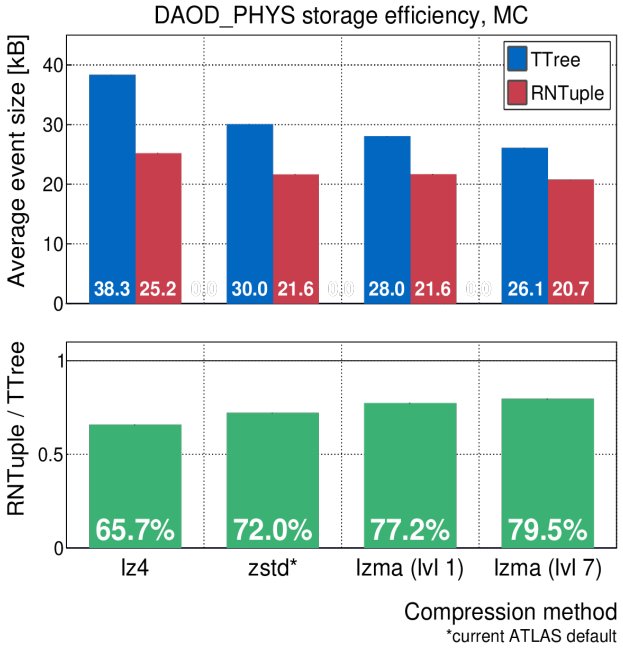
Day-to-day support via **Mattermost channel**

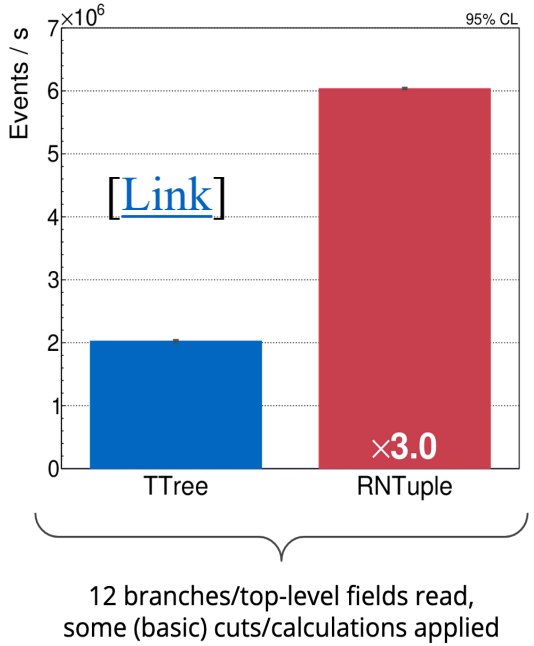


+ whole of LHCb community as we progress with Run III

Future work: Funtuple with RNTuple

Active effort to integrate FunTuple with RNTuple (**thread-safe** and can save **20-35% of storage** space with **factor 3 increase in throughput)**.





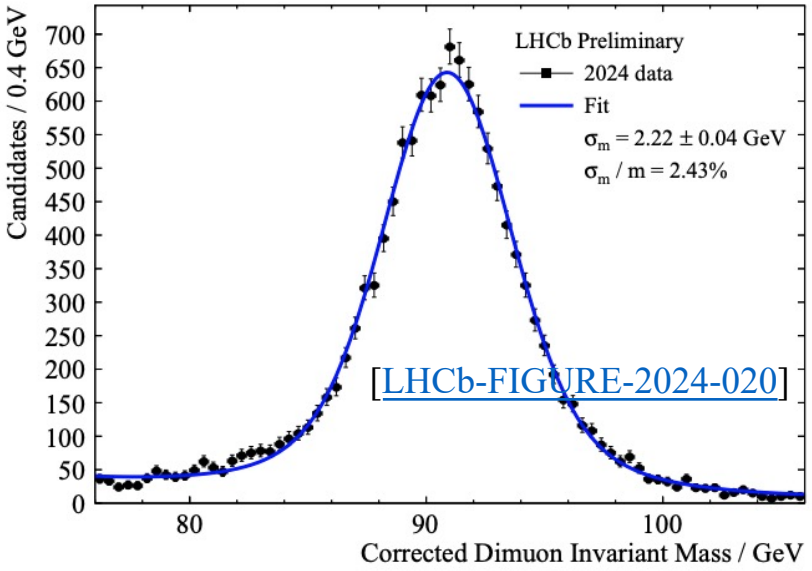
See separate talk by Silia Taider.

Summary and conclusions

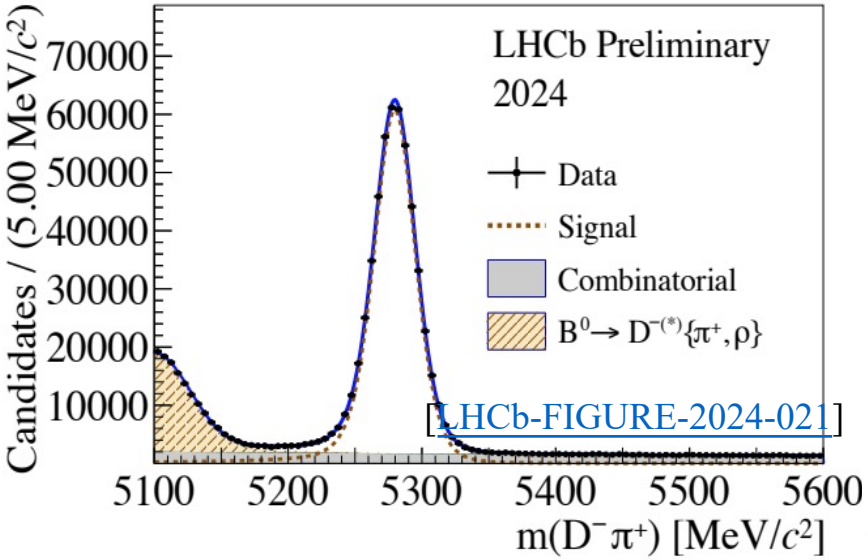
Presented a summary of offline data processing for LHCb Run 3, focusing on FunTuple: past, present, and future.

Exciting results from Run 3 analysis coming soon!

$Z \rightarrow \mu^+\mu^-$



$B^0 \rightarrow D^-\pi^+$



Luminosity