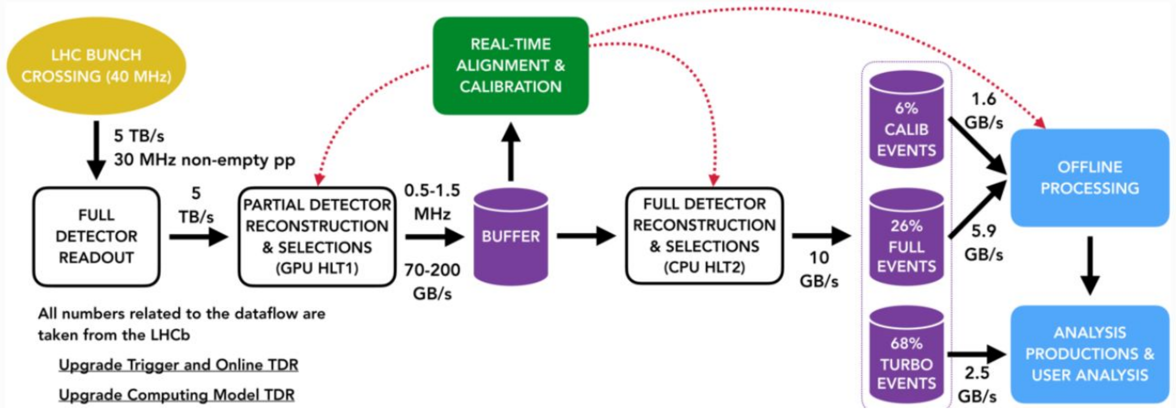

GUARDIANS
OF THE DAVINCI

Format

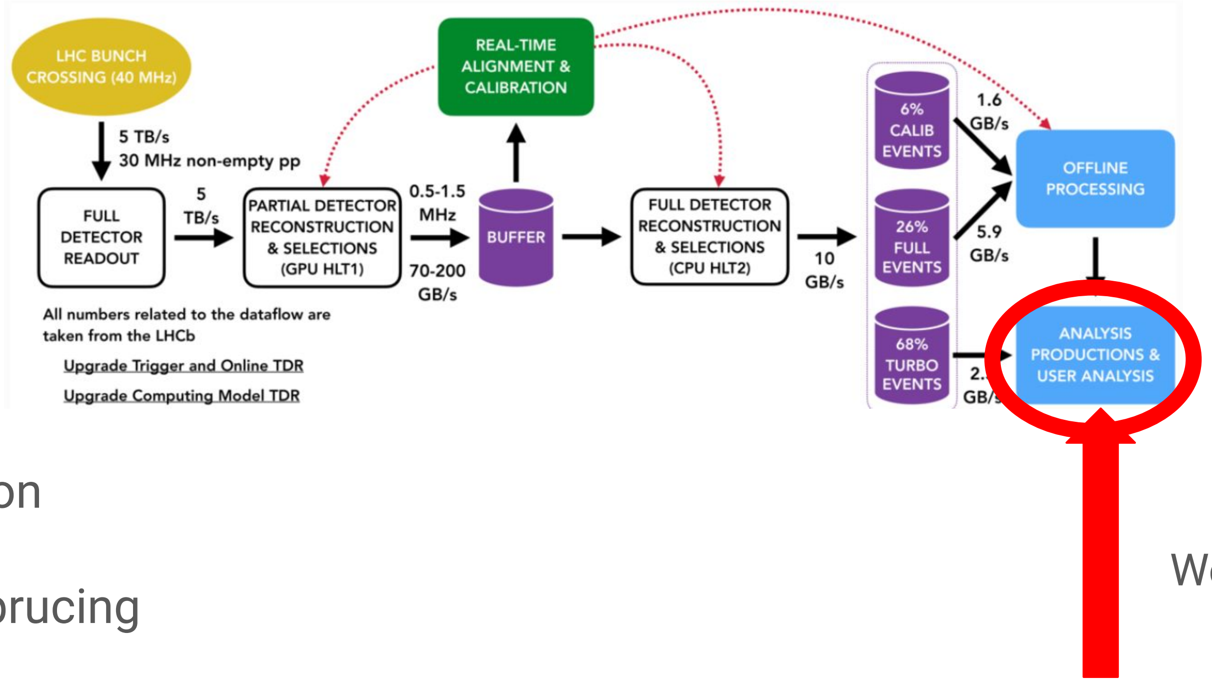
- These DaVinci lessons will be given with some slides to explain the ideas but also **hands on** running the software!
- If you have a question please just ask! One of us here should be able to help!
- These slides and gitlab with the examples will be uploaded to the indico!
- You have looked at DaVinci for Run1/2 on Tuesday - some bits may look similar!
- I don't expect you to remember everything! This should hopefully be a resource you can look back on + the gitlab!!
- **Note: I am not an expert, I've just been using it for the year!!**

Where are we?



LHC BUNCH CROSSING (40 MHz)
5 TB/s
30 MHz non-empty pp
FULL DETECTOR READOUT
5 TB/s
PARTIAL DETECTOR RECONSTRUCTION & SELECTIONS (GPU HLT1)
0.5-1.5 MHz
70-200 GB/s
REAL-TIME ALIGNMENT & CALIBRATION
BUFFER
FULL DETECTOR RECONSTRUCTION & SELECTIONS (CPU HLT2)
10 GB/s
6% CALIB EVENTS
26% FULL EVENTS
68% TURBO EVENTS
1.6 GB/s
5.9 GB/s
2.5 GB/s
OFFLINE PROCESSING
ANALYSIS PRODUCTIONS & USER ANALYSIS
All numbers related to the dataflow are taken from the LHCb
Upgrade Trigger and Online TDR
Upgrade Computing Model TDR

Where are we?



1. PP collision
2. HLT1
3. HLT2 + Sprucing
4. DaVinci

What is DaVinci?

- DaVinci processes the output from HLT2/Sprucing (stored in DST files) into compact ROOT files for analysis.
- The three main streams from HLT2 are:
  - Turbo Lines
  - Full Stream Lines
  - Calibration Lines

DaVinci GitHub here

HLT2 Lines

- Turbo Lines:
  - For exclusive decays with fully specified final states.
  - Saves trigger candidates and their descendants .
  - Creates "physics-ready" candidates directly.

- Full Stream Lines:
  - For inclusive decay searches.
  - Saves all reconstructed tracks from events passing loose topological triggers or selections.
  - Enables multiple analyses to use the same base selection.

- Calibration Lines:
  - High-rate, clean decay modes.
  - Persists complete information (including RAW banks).
  - Used for efficiency measurements and detector calibration.

Sprucing 🌲

- Sprucing processes each stream differently:

- Turbo Stream:
  - "Sprucing Passthrough" reorganizes data structure.
  - Preserves physics content.
  - Ensures consistency with full stream format.

- Full Stream:
  - Performs "slimming and skimming".
  - Enables specific decay searches using common trigger selections.
  - More efficient than individual turbo lines for inclusive analyses.
  - Creates “pruned”, analysis-ready output.

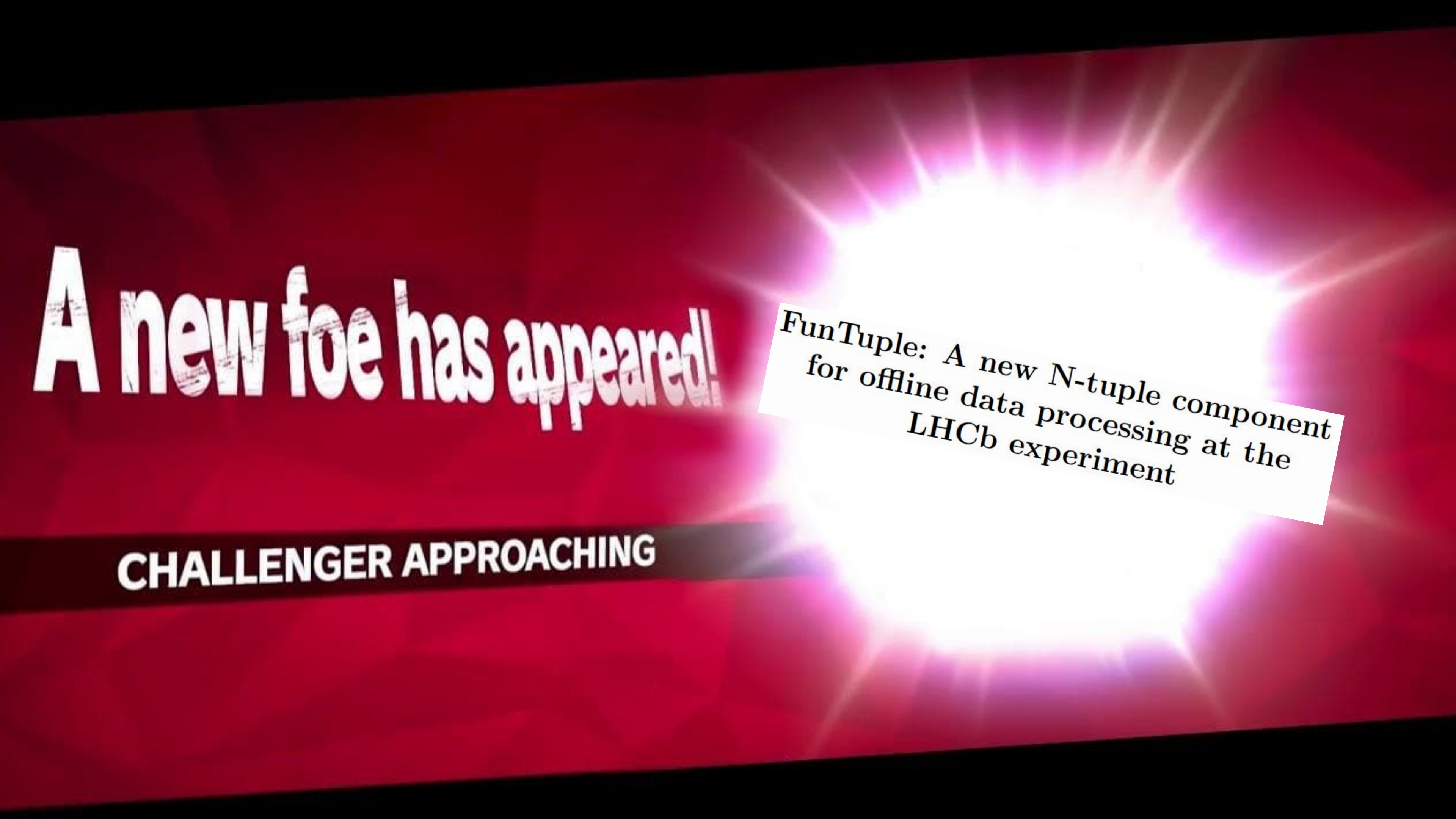
DaVinci Run 3

- Moving from Run 2 to Run 3 there would be some nice changes…

- Would be good if the DaVinci selection framework was the same as the online framework - it would ensure consistency across computations (Basically HLT2/Sprucing/DaVinci all using the same variable calculations - “Functors”).

- Would be good if we had more flexibility to what values we wanted to calculate in our tuples….


A new foe has appeared!
CHALLENGER APPROACHING


A new foe has appeared!
CHALLENGER APPROACHING
FunTuple: A new N-tuple component for offline data processing at the LHCb experiment

DaVinci Run 3

In comes FunTuple (**Fun**ctional n**Tuple**) replacing the old DecayTreeTuple providing these needed changes.

ThOr Functors provide these calculations!

FunTuple Paper Link - Lots of cool info on how this works !

The Key Takeaway 🥘:

As a user DaVinci makes your root files with the variables you want after the reconstruction / sprucing stages.

The Setup

The Set Up

- Let's say you are asked to produce a root file for your decay mode of B+ ->J/psi K+ .
- You will probably want to know the mass of these reconstructed particles, their energies, how likely they are to be that certain type particle etc.
- Otherwise how can you analyse this decay…
- This is easily configurable with DaVinci !

The Set Up 2

- There are two scripts needed to run DaVinci:
  1. A python file 🐍 where you say what decay you are looking at and what variables you want it to calculate for you.
  2. A yaml file to tell it what is the input file, output file, detector conditions, if it's simulation etc -> configuration file.

  (don't panic a .yaml file is basically a fancy text file)

The Basic Python Script

```python
from PyConf.reading import get_particles, get_pvs
from RecoConf.event_filters import require_pvs
from DaVinci import make_config, Options
from DaVinci.algorithms import create_lines_filter
from FunTuple import FunTuple_Particles as Funtuple
from FunTuple import FunctorCollection
import FunTuple.functorcollections as FC
import Functors as F

def main(options: Options):
    line = "Hlt2B2CC_BuToJpsiKplus_JpsiToMuMu_Detached"
    data = get_particles(f"/Event/HLT2/{line}/Particles")
    line_prefilter = create_lines_filter(name=f"PreFilter_{line}", lines=[line])
    pvs = get_pvs()

    fields = {
        "Bplus": "[B+ -> (J/psi(1S) -> mu+ mu-) K+]CC",
        "Jpsi": "[B+ -> ^(J/psi(1S) -> mu+ mu-) K+]CC",
        "muplus": "[B+ -> (J/psi(1S) -> ^mu+ mu-) K+]CC",
        "muminus": "[B+ -> (J/psi(1S) -> mu+ ^mu-) K+]CC",
        "Kplus": "[B+ -> (J/psi(1S) -> mu+ mu-) ^K+]CC",
    }

    all_vars = FunctorCollection({
        "M": F.MASS,
        "P": F.P,
        "PT": F.PT
    })

    variables = {"ALL": all_vars}

    funtuple = Funtuple(
        name=line,
        tuple_name="DecayTree",
        fields=fields,
        variables=variables,
        inputs=data,
    )

    algs = {line: [line_prefilter, require_pvs(pvs), funtuple]}
    return make_config(options, algs)
```

Some imports

```python
from PyConf.reading import get_particles, get_pvs
from RecoConf.event_filters import require_pvs
from DaVinci import make_config, Options
from DaVinci.algorithms import create_lines_filter
from FunTuple import FunTuple_Particles as Funtuple
from FunTuple import FunctorCollection
import FunTuple.functorcollections as FC
import Functors as F

def main(options: Options):
    line = "Hlt2B2CC_BuToJpsiKplus_JpsiToMuMu_Detached"
    data = get_particles(f"/Event/HLT2/{line}/Particles")
    line_prefilter = create_lines_filter(name=f"PreFilter_{line}", lines=[line])
    pvs = get_pvs()

    fields = {
        "Bplus": "[B+ -> (J/psi(1S) -> mu+ mu-) K+]CC",
        "Jpsi": "[B+ -> ^(J/psi(1S) -> mu+ mu-) K+]CC",
        "muplus": "[B+ -> (J/psi(1S) -> ^mu+ mu-) K+]CC",
        "muminus": "[B+ -> (J/psi(1S) -> mu+ ^mu-) K+]CC",
        "Kplus": "[B+ -> (J/psi(1S) -> mu+ mu-) ^K+]CC",
    }

    all_vars = FunctorCollection({
        "M": F.MASS,
        "P": F.P,
        "PT": F.PT
    })

    variables = {"ALL": all_vars}

    funtuple = Funtuple(
        name=line,
        tuple_name="DecayTree",
        fields=fields,
        variables=variables,
        inputs=data,
    )

    algs = {line: [line_prefilter, require_pvs(pvs), funtuple]}
    return make_config(options, algs)
```

Some imports

```python
from PyConf.reading import get_particles, get_pvs
from RecoConf.event_filters import require_pvs
from DaVinci import make_config, Options
from DaVinci.algorithms import create_lines_filter
from FunTuple import FunTuple_Particles as Funtuple
from FunTuple import FunctorCollection
import FunTuple.functorcollections as FC
import Functors as F

def main(options: Options):
    line = "Hlt2B2CC_BuToJpsiKplus_JpsiToMuMu_Detached"
    data = get_particles(f"/Event/HLT2/{line}/Particles")
    line_prefilter = create_lines_filter(name=f"PreFilter_{line}", lines=[line])
    pvs = get_pvs()

    fields = {
        "Bplus": "[B+ -> (J/psi(1S) -> mu+ mu-) K+]CC",
        "Jpsi": "[B+ -> ^(J/psi(1S) -> mu+ mu-) K+]CC",
        "muplus": "[B+ -> (J/psi(1S) -> ^mu+ mu-) K+]CC",
        "muminus": "[B+ -> (J/psi(1S) -> mu+ ^mu-) K+]CC",
        "Kplus": "[B+ -> (J/psi(1S) -> mu+ mu-) ^K+]CC",
    }

    all_vars = FunctorCollection({
        "M": F.MASS,
        "P": F.P,
        "PT": F.PT
    })

    variables = {"ALL": all_vars}

    funtuple = Funtuple(
        name=line,
        tuple_name="DecayTree",
        fields=fields,
        variables=variables,
        inputs=data,
    )

    algs = {line: [line_prefilter, require_pvs(pvs), funtuple]}
    return make_config(options, algs)
```

Define our main function, define our line, tell it where the events are stored in the dst and create a prefilter.

Some imports

```python
from PyConf.reading import get_particles, get_pvs
from RecoConf.event_filters import require_pvs
from DaVinci import make_config, Options
from DaVinci.algorithms import create_lines_filter
from FunTuple import FunTuple_Particles as Funtuple
from FunTuple import FunctorCollection
import FunTuple.functorcollections as FC
import Functors as F

def main(options: Options):
    line = "Hlt2B2CC_BuToJpsiKplus_JpsiToMuMu_Detached"
    data = get_particles(f"/Event/HLT2/{line}/Particles")
    line_prefilter = create_lines_filter(name=f"PreFilter_{line}", lines=[line])
    pvs = get_pvs()

    fields = {
        "Bplus": "[B+ -> (J/psi(1S) -> mu+ mu-) K+]CC",
        "Jpsi": "[B+ -> ^(J/psi(1S) -> mu+ mu-) K+]CC",
        "muplus": "[B+ -> (J/psi(1S) -> ^mu+ mu-) K+]CC",
        "muminus": "[B+ -> (J/psi(1S) -> mu+ ^mu-) K+]CC",
        "Kplus": "[B+ -> (J/psi(1S) -> mu+ mu-) ^K+]CC",
    }

    all_vars = FunctorCollection({
        "M": F.MASS,
        "P": F.P,
        "PT": F.PT
    })

    variables = {"ALL": all_vars}

    funtuple = Funtuple(
        name=line,
        tuple_name="DecayTree",
        fields=fields,
        variables=variables,
        inputs=data,
    )

    algs = {line: [line_prefilter, require_pvs(pvs), funtuple]}
    return make_config(options, algs)
```

Define our main function, define our line, tell it where the events are stored in the dst and create a prefilter.

Define our decay

Some imports

```python
from PyConf.reading import get_particles, get_pvs
from RecoConf.event_filters import require_pvs
from DaVinci import make_config, Options
from DaVinci.algorithms import create_lines_filter
from FunTuple import FunTuple_Particles as Funtuple
from FunTuple import FunctorCollection
import FunTuple.functorcollections as FC
import Functors as F

def main(options: Options):
    line = "Hlt2B2CC_BuToJpsiKplus_JpsiToMuMu_Detached"
    data = get_particles(f"/Event/HLT2/{line}/Particles")
    line_prefilter = create_lines_filter(name=f"PreFilter_{line}", lines=[line])
    pvs = get_pvs()

    fields = {
        "Bplus": "[B+ -> (J/psi(1S) -> mu+ mu-) K+]CC",
        "Jpsi": "[B+ -> ^(J/psi(1S) -> mu+ mu-) K+]CC",
        "muplus": "[B+ -> (J/psi(1S) -> ^mu+ mu-) K+]CC",
        "muminus": "[B+ -> (J/psi(1S) -> mu+ ^mu-) K+]CC",
        "Kplus": "[B+ -> (J/psi(1S) -> mu+ mu-) ^K+]CC",
    }

    all_vars = FunctorCollection({
        "M": F.MASS,
        "P": F.P,
        "PT": F.PT
    })

    variables = {"ALL": all_vars}

    funtuple = Funtuple(
        name=line,
        tuple_name="DecayTree",
        fields=fields,
        variables=variables,
        inputs=data,
    )

    algs = {line: [line_prefilter, require_pvs(pvs), funtuple]}
    return make_config(options, algs)
```

Define our main function, define our line, tell it where the events are stored in the dst and create a prefilter.

Define our decay

Pick what values we want calculated for each particle.

Some imports

```python
from PyConf.reading import get_particles, get_pvs
from RecoConf.event_filters import require_pvs
from DaVinci import make_config, Options
from DaVinci.algorithms import create_lines_filter
from FunTuple import FunTuple_Particles as Funtuple
from FunTuple import FunctorCollection
import FunTuple.functorcollections as FC
import Functors as F
```

Define our main function, define our line, tell it where the events are stored in the dst and create a prefilter.

```python
def main(options: Options):
    line = "Hlt2B2CC_BuToJpsiKplus_JpsiToMuMu_Detached"
    data = get_particles(f"/Event/HLT2/{line}/Particles")
    line_prefilter = create_lines_filter(name=f"PreFilter_{line}", lines=[line])
    pvs = get_pvs()
```

Define our decay

```python
    fields = {
        "Bplus": "[B+ -> (J/psi(1S) -> mu+ mu-) K+]CC",
        "Jpsi": "[B+ -> ^(J/psi(1S) -> mu+ mu-) K+]CC",
        "muplus": "[B+ -> (J/psi(1S) -> ^mu+ mu-) K+]CC",
        "muminus": "[B+ -> (J/psi(1S) -> mu+ ^mu-) K+]CC",
        "Kplus": "[B+ -> (J/psi(1S) -> mu+ mu-) ^K+]CC",
    }
```

Pick what values we want calculated for each particle.

```python
    all_vars = FunctorCollection({
        "M": F.MASS,
        "P": F.P,
        "PT": F.PT
    })

    variables = {"ALL": all_vars}
```

Set up funtuple and chose your algorithms!

```python
    funtuple = Funtuple(
        name=line,
        tuple_name="DecayTree",
        fields=fields,
        variables=variables,
        inputs=data,
    )

    algs = {line: [line_prefilter, require_pvs(pvs), funtuple]}
    return make_config(options, algs)
```

Yaml

Data yaml


```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/Data1.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: False
input_process: "TurboPass"
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: True
data_type: "Upgrade"
evt_max: 1000
print_freq: 1000
ntuple_file: Data.root
```


MC yaml


```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```


The addition of the dddb_tag and conddb_tag are for det-desc builds.

Yaml

This is a .dst I have copied from bookkeeping to my /eos space.

```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

Yaml

Even though the input is a dst file these are a “type” of ROOT file so both input and output here are ROOT.

```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

Yaml

This is just a given number for the input type e.g. dst and root are all 0.5, other file types differ.

```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

Yaml

Is your sample simulation?

```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

Yaml

What is the input process coming from e.g. hlt2 or sprucing for MC or TurboPass etc for data.

```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

Yaml

What working group is it from?
e.g. QEE, BnoC, BandQ etc.

```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

Yaml

The geometry tag specifies things such as details about the shapes, sizes, positions, and materials of each detector element (time independent factors).

Conditions tell you about the calibration, alignment and things like temperatures (time dependent factors).

More info here

```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

Yaml

If you want the luminosity info -> comes in a separate tree called Lumitree.

The data_type is the "Upgraded" detector

```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

Yaml

evt_max tells it how many events to run over =-1 does all.

print_freq tells it how often to print how far it has processed.

```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

Yaml

Where do you want the file output and what do you want it called.

```yaml
input_files:
  - /eos/lhcb/user/g/ghallett/sk/MC.dst
input_type: ROOT
output_type: ROOT
input_raw_format: 0.5
simulation: True
input_process: Hlt2
input_stream: b2cc
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
lumi: False
data_type: "Upgrade"
evt_max: 1000
print_freq: 100
ntuple_file: MC.root

# These tags are required if you run over a det-desc build (rather then DD4HEP)
# Depends on what conditions your simulation was requested in
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

Get a Wriggle On

- Lets stop mucking around and get coding…
- This tutorial will be run on lxplus. If you are following along on a local university cluster make sure you source the correct environment first!

(source /cvmfs/lhcb.cern.ch/lib/LbEnv and you will need to access my files on /eos -> for the lesson please use lxplus)

- Step 1: Run this basic script on MC
- All code with ***extensive*** comments are on the gitlab here so you can just copy these if you wish…
- But I will type them out here (without comments of course and depending on time!) in person for the lesson so please follow along!

The run command

- To run this:

lb-run -c best (or specify platform I used: x86_64_v3-el9-gcc13-opt+g) DaVinci/latest or (DaVinci/v64r12 I used) lbexec DV_basic:main MC/data_options.yaml

- So lb-run sets up the software environment
- -c is the flag to set the platform
- DaVinci/version means you can pick which release of DaVinci you want
- lbexec does some more environment setup
- DV_basic:main is our DaVinci python file with this :main to point to our main function
- Finally the options.yaml for either data or MC.

Note: Other platforms are available from all good retailers…

- For those not following at the workshop if you have run correctly it should start DaVinci and look like this:

```
==============================================================================================================================
                                                   Welcome to DaVinci version 64.12
                                          running on lxplus927.cern.ch on Fri Nov  1 13:07:54 2024
==============================================================================================================================
ApplicationMgr       INFO Application Manager Configured successfully
NTupleSvc                            INFO Added stream file:../Starterkit_2024/MC.root as FILE1
HLTControlFlowMgr                    INFO Start initialization
RootHistSvc                          INFO Writing ROOT histograms to: ../Starterkit_2024/MC.root
HistogramPersistencySvc              INFO Added successfully Conversion service:RootHistSvc
Hlt2B2CC_BuToJpsiKplus_JpsiToMuM...  INFO User specified descriptor: [B+ -> (J/psi(1S) -> mu+ mu-) K+]CC
Hlt2B2CC_BuToJpsiKplus_JpsiToMuM...  INFO Number of decay possibilities with specified descriptor: 2
Hlt2B2CC_BuToJpsiKplus_JpsiToMuM...  INFO Possibility #0: B+ -> (J/psi(1S) -> mu+ mu-) K+
Hlt2B2CC_BuToJpsiKplus_JpsiToMuM...  INFO Possibility #1: B- -> (J/psi(1S) -> mu- mu+) K-
HLTControlFlowMgr                    INFO Concurrency level information:
HLTControlFlowMgr                    INFO  o Number of events slots: 1
HLTControlFlowMgr                    INFO  o TBB thread pool size:  'ThreadPoolSize':1
HLTControlFlowMgr                    INFO ---> End of Initialization. This took 263 ms
ApplicationMgr                       INFO Application Manager Initialized successfully
FunctorFactory                       INFO Reusing functor library: "/tmp/ghallett/FunctorJitLib_0xf4f1e73418b3f303_0x90cd19fdc47a33
ApplicationMgr                       INFO Application Manager Started successfully
EventPersistencySvc                  INFO Added successfully Conversion service:RootCnvSvc
EventSelector                        INFO Stream:EventSelector.DataStreamTool_1 Def:DATAFILE='/eos/lhcb/user/g/ghallett/sk/MC.dst'
'READ' IgnoreChecksum='YES'
HLTControlFlowMgr                    INFO Will measure time between events 1000 and 9000 (stop might be some events later)
HLTControlFlowMgr                    INFO Starting loop on events
EventSelector                     SUCCESS Reading Event record 1. Record number within stream 1: 1
PersistencyIO       INFO  +++ Set Streamer to dd4hep::OpaqueDataBlock
DD4hep              INFO  ++ Using globally Geant4 unit system (mm,ns,MeV)
```

- Then a load of DD4HEP detector setup:

```
Compact                     INFO ++ Already processed xml document file:/cvmfs/lhcb.cern.ch/lib/lhcb/DETECTOR/DETECTOR_v1r
als.xml.
Compact                     INFO ++ Already processed xml document file:/cvmfs/lhcb.cern.ch/lib/lhcb/DETECTOR/DETECTOR_v1r
als.xml.
Compact                     INFO ++ Converted subdetector:BlockWallBefMagnet of type DD4hep_VolumeAssembly
Compact                     INFO ++ Converted subdetector:BlockWallUpStr of type DD4hep_VolumeAssembly
Compact                     INFO ++ Converted subdetector:GValve of type DD4hep_VolumeAssembly
Compact                     INFO ++ Converted subdetector:Bls of type LHCb_Bls_v1_0 [sensitive]
Compact                     INFO ++ Converted subdetector:BcmUp of type LHCb_Bcm_v1_0 [sensitive]
Compact                     INFO ++ Converted subdetector:BcmDown of type LHCb_Bcm_v1_0 [sensitive]
Compact                     INFO ++ Converted subdetector:MBXWUp of type LHCb_MBXW_v1_0
Compact                     INFO ++ Converted subdetector:MBXWSUp of type LHCb_MBXW_v1_0
Compact                     INFO ++ Converted subdetector:MBXWDown of type LHCb_MBXW_v1_0
Compact                     INFO ++ Converted subdetector:PipeUpstream of type DD4hep_VolumeAssembly
Compact                     INFO ++ Converted subdetector:VMAAAUpstreamVax of type LHCb_Pipe_VMA_v1_0
Compact                     INFO ++ Converted subdetector:VMAAAUpstream of type LHCb_Pipe_VMA_v1_0
Compact                     INFO ++ Converted subdetector:VMACAUpstream1 of type LHCb_Pipe_VMA_v1_0
Compact                     INFO ++ Converted subdetector:VMACAUpstream2 of type LHCb_Pipe_VMA_v1_0
Compact                     INFO ++ Converted subdetector:VMABKUpstream of type LHCb_Pipe_VMA_v1_0
Compact                     INFO ++ Converted subdetector:Rich1 of type LHCb_Rich1_Geometry_RUN3_v1
Compact                     INFO ++ Converted subdetector:VP of type LHCb_VP_v1_0 [tracker]
Compact                     INFO ++ Converted subdetector:UT of type LHCb_UT_v1_0 [tracker]
Compact                     INFO ++ Converted subdetector:PipeBeforeVelo of type DD4hep_VolumeAssembly
Compact                     INFO ++ Converted subdetector:PipeBeforeVeloSupFix of type DD4hep_VolumeAssembly
Compact                     INFO ++ Converted subdetector:PipeBeforeMagnet of type DD4hep_VolumeAssembly
Compact                     INFO ++ Converted subdetector:VPUpStreamPipe of type DD4hep_VolumeAssembly
Compact                     INFO ++ Converted subdetector:VeloDownStreamPipe of type DD4hep_VolumeAssembly
Compact                     INFO ++ Converted subdetector:PipeInUT of type DD4hep_VolumeAssembly
```

DD4HEP is just the software framework the detector is modelled in!

- Then the print outs from reading the events out (at whatever interval you set the print_freq):

```
HLTControlFlowMgr                       INFO Timing started at: 13:08:12
EventSelector                        SUCCESS Reading Event record 1001. Record number within stream 1: 1001
EventSelector                        SUCCESS Reading Event record 2001. Record number within stream 1: 2001
EventSelector                        SUCCESS Reading Event record 3001. Record number within stream 1: 3001
EventSelector                        SUCCESS Reading Event record 4001. Record number within stream 1: 4001
EventSelector                        SUCCESS Reading Event record 5001. Record number within stream 1: 5001
EventSelector                        SUCCESS Reading Event record 6001. Record number within stream 1: 6001
EventSelector                        SUCCESS Reading Event record 7001. Record number within stream 1: 7001
EventSelector                        SUCCESS Reading Event record 8001. Record number within stream 1: 8001
HLTControlFlowMgr                       INFO Timing stopped at: 13:09:02
EventSelector                        SUCCESS Reading Event record 9001. Record number within stream 1: 9001
HLTControlFlowMgr                       INFO ---> Loop over 10000 Events Finished -  WSS 2105.79, timed 8000 Events: 49273 ms, Evts/s = 162.361
FilterDstDataSize                       INFO Number of counters : 1
```

- Then you can see the execution of algorithms to see where events were lost e.g. from the prefilter, requiring pvs or the actual line.
- This will really help for debugging if something goes wrong!

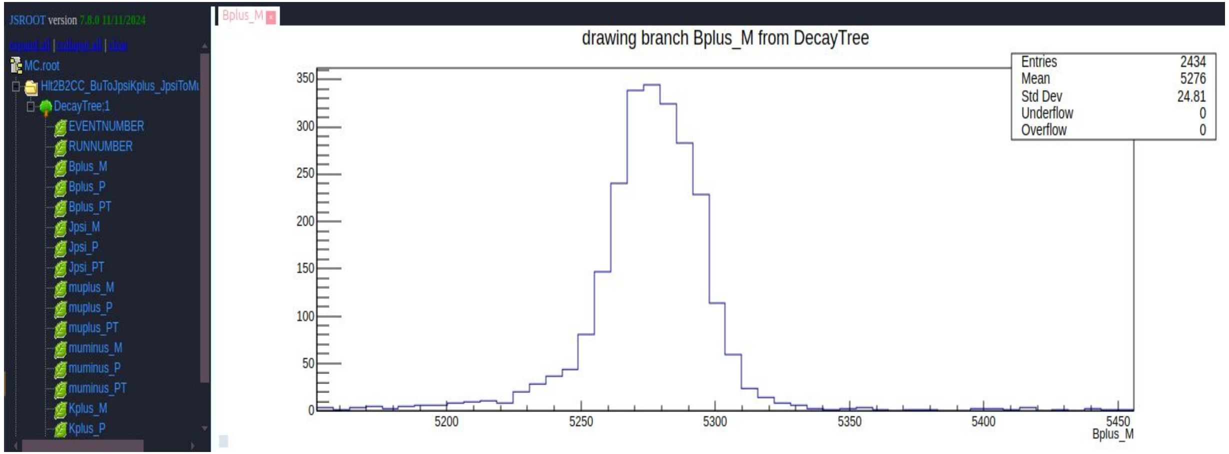
```
HLTControlFlowMgr                      INFO StateTree: CFNode   #executed  #passed
LAZY_AND: DaVinci                                                 #=10000    Sum=2389         Eff=|( 23.89000 +- 0.426412)%|
 NONLAZY_OR: UserAnalysis                                         #=10000    Sum=2389         Eff=|( 23.89000 +- 0.426412)%|
  LAZY_AND: Hlt2B2CC_BuToJpsiKplus_JpsiToMuMu_Detached            #=10000    Sum=2389         Eff=|( 23.89000 +- 0.426412)%|
   RawBankSizeFilter/FilterDstDataSize                            #=10000    Sum=10000        Eff=|( 100.0000 +- 0.00000 )%|
   VoidFilter/PreFilter_Hlt2B2CC_BuToJpsiKplus_JpsiToMuMu_Detached #=10000    Sum=2389         Eff=|( 23.89000 +- 0.426412)%|
   VoidFilter/require_pvs                                         #=2389     Sum=2389         Eff=|( 100.0000 +- 0.00000 )%|
   FunTupleBase_Particles/Hlt2B2CC_BuToJpsiKplus_JpsiToMuMu_Detached #=2389  Sum=2389         Eff=|( 100.0000 +- 0.00000 )%|
```

- Then right at the end it should say it was finalised and terminated successfully!

```
HLTControlFlowMgr                      INFO Histograms converted successfully according to request.
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-199]                age:  0 [40829 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-229999]             age:  0 [    2 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-232943]             age:  0 [    2 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-233999]             age:  0 [    1 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-236999]             age:  0 [    2 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-246236]             age:  0 [1369 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-248999]             age:  0 [ 260 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-254301]             age:  0 [    1 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-254999]             age:  0 [    4 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-255799]             age:  0 [    2 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-267949]             age:  0 [ 122 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-270999]             age:  0 [    1 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-272086]             age:  0 [    1 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-274699]             age:  0 [   13 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-291592]             age:  0 [   14 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-291999]             age:  0 [1183 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-294033]             age:  0 [    1 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-294999]             age:  0 [    1 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-295401]             age:  0 [    3 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-297999]             age:  0 [    4 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-304424]             age:  0 [    2 entries]
ConditionsPool                         INFO +++ Remove Conditions for pool with IOV: run(0):[0-9223372036854775807] age:  0 [   59 entries]
ToolSvc                                INFO Removing all tools created by ToolSvc
RootCnvSvc                             INFO Disconnected data IO:37D0780A-9620-11EF-8A35-000000C9FE80 [/eos/lhcb/user/g/ghallett/sk/MC.dst]
NTupleSvc                              INFO NTuples saved successfully
ApplicationMgr                         INFO Application Manager Finalized successfully
ApplicationMgr                         INFO Application Manager Terminated successfully
[ghallett@lxplus927 davincirun3_starterkit2024]$
```

What's in it?

- Now you can checkout your root file however you like. Personally I often use a vscode extension called "root file viewer" but it's up to you! It should look something like this:



By the convention, the LHCb default units are MeV, millimeters and nanoseconds.

If this worked you are now a DaVinci expert congrats

🍹

- This brings us to the end of the first lesson on running a basic script!
- Please enjoy the coffee break and if you have any questions please come and ask one of us!
- We will continue after the break with Functors!!!

BACKUP

Note on ThOr Functors

This component is configured with a robust suite of tools designed for the second stage of the LHCb trigger system, known as Throughput Oriented (ThOr) functors. These functors are designed to deliver high-speed in the trigger's demanding throughput environment and are adept at computing topological and kinematic observables.

What are the platforms?

- x86_64_v3-el9-gcc13-opt+g =

- x86_64_v3 -> CPU architecture
- el9 -> operating system (LINUX 9 here)
- gcc13 -> the compiler
- opt -> optimisation level
- +g -> debug symbols included

Turbo - Full - Calibration (Credit conversation with the man the myth the legend Luke Grazette - this is me reiterating what he just told me so any errors here are my own lol)

- Ok so turbo lines are exclusive decays where we fully specify the tracks we are looking for e.g. Lambda b -> Lambda->(p pi) K+ K-. After turbo we are left with "physics ready" candidates. The Sprucing passthrough doesn't change any physics (the data structure is reorganised to match that of the output of full sprucing). The purpose of full stream is for inclusive decays so at the HLT2 level you might have topological triggers that e.g. might look for a 2/3 body combo with some loose kinematics, then all the reconstructed tracks (not rawbanks) are saved for the event that passed the topo requirement. Then in Sprucing specific decays can use the same topoline for their specific search (since we saved all tracks from that event). Another example being your Hlt2SingleHighPtMuon line, lots of decays need this requirement so if you wrote them all individually for turbo it would be more expensive as all the selectively persisted objects are saved accessibly via the grid rather than the "pruned" output from sprucing. The calibration stream tend to be clean/high rate decay modes that we can use to evaluate efficiencies and mis-calibrations from the detector. Most of these lines fully persist all the information (including rawbanks) so they can re-perform reconstruction.

More on Geometry and Conditions tags

-

Overview

This page collects all information about the geometry description and conditions of the BGV detector.

The Detector Description in Gaudi (see Chapter 8 of the Gaudi User Guide) encompasses the structure, geometry, material and other properties of the detector or any other element of the given setup. In practice, clear distinction is made between the following parts of the Detector Description:

- **Geometry description**, i.e. structure, geometry and material of all objects (sensitive or not) making up a given detector setup. *Sometimes we will refer to this part as DDDB*, Detector Description Database (this is just an alias; strictly speaking it is not a "correct" alias, because the "detector description" includes all properties not only the geometry-related ones, and because the implementation can be with xml files and not necessarily with a database)
- **Conditions**, i.e. constant or time-varying properties related to the detector or its surrounding (e.g. ModuleID - TELL1 mapping, module spatial alignment constants, detector channel pedestals, temperature in the tunnel, etc.). *Sometimes we will refer to this part as CondDB*. It is possible the experiment control system to generate conditions *Online*, and store them in dedicated files

If you wish to copy the files yourself I used these:

- The bkk for the Data is: /LHCb/Collision24/Beam6800GeV-VeloClosed-MagDown/Real Data/Sprucing24c4/94000000/B2CC.DST
- For MC: /MC/2024/Beam6800GeV-2024.W31.34-MagUp-Nu6.3-25ns-Pythia8/Sim10d /HLT2-2024.W31.34/12143001/DST