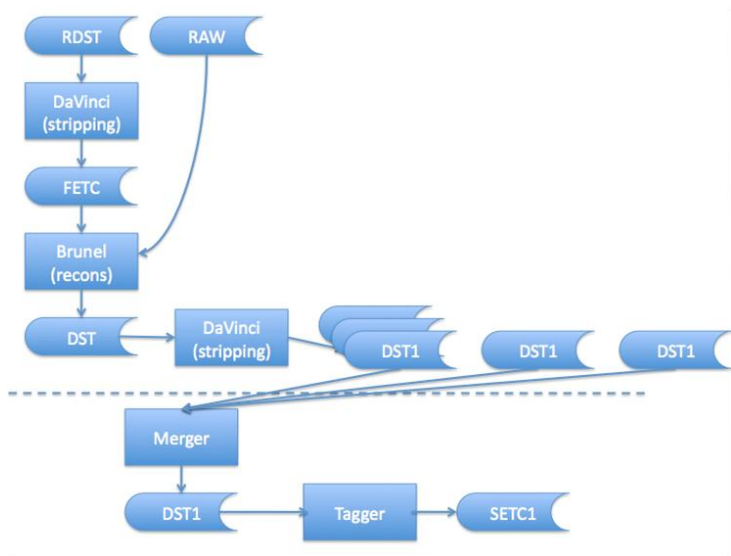


Stripping framework

Anton Poluektov

The University of Warwick, UK,
Budker Institute of Nuclear Physics, Novosibirsk, Russia

Stripping workflow



- RDST → FETC
Implemented.
- DST → DST streams
implemented. Currently, Gaudi can handle not more than 5 DST streams.
- DST → SETC (tagger).
Look for TES locations corresponding to selections, and fill ETC. No need to run selections.
Will be released in DaVinci v24r3p1.
Implemented by appending TESChecker instead of StrippingAlg to stream sequencers. But TES container (/Event/Strip/Phys/[Selection]) is there more often than the corresponding selection triggers. Need to look if the container is not empty ⇒ dedicated algorithm?

- MC09 stripping (MC09 DST → FETC → DST → stream DSTs)
 - Remaining issues with running L0 and HLT1 are solved by Juan.
 - Bookkeeping cannot yet handle the new file types (being fixed).
 - Otherwise ready to launch. 200 jobs submitted yesterday by Stuart, but one of them failed with segfault.
- "Full" stripping (rDST → FETC → DST → stream DSTs)
 - Need a patch release of DaVinci to handle rDST (v24r3p1).

Consider the sequencer that does $B \rightarrow Dh$ selection is there (B2DhFilterSequence). Then:

```
line = StrippingLine('B2Dh'  
    , prescale = 1  
    , algos = [ B2DhFilterSequence ]  
    , postscale = 1  
    , stream = 'BExclusive'  
)
```

will integrate the selection into stripping framework. This will create a line corresponding to B2DhFilterSequence selection, and add it to the list of lines. The selection will be assigned to the stream "BExclusive".

Selections are stored in `STRIPPINGSELECTIONSROOT/options/`. The new selection has to be added (via `importOptions`) to `STRIPPINGSELECTIONSROOT/options/StrippingSelections.py` (this is the main file loaded by default).

StrippingLine can take StrippingMember in the list of algorithms. StrippingMember is implemented similarly to Hlt2Member. Cloning of lines may be convenient for e. g. signal and sidebands selections. Example:

```
combine = CombineParticles("Bd2KstarMuMuCombine")

filter = StrippingMember (FilterDesktop
    , "Filter"
    , InputLocations = ["Bd2KstarMuMuCombine"]
    , Code = "BLABLA_1"
)

line1 = StrippingLine('B2DPi_signal'
    , prescale = 1
    , algos = [ combine , filter ]
    , stream = 'BExclusive'
)

line2 = line1.clone('B2Dpi_sideband'
    , prescale = 0.2
    , FilterDesktopFilter = {"Code" : "BLABLA_2"}
)
```

StrippingLine can take SelectionSequence in the list of algorithms. SelectionSequence needs to be put to python

```
name = "Bs2JpsiPhi"

...

Bs = Selection ( "Sel"+name,
                Algorithm = _Bs,
                RequiredSelections = [Jpsi, Phi])
Bs.__apply_configuration__()

sequence = SelectionSequence("Seq"+name, TopSelection = Bs)
sequence.__apply_configuration__()
```

In options/StrippingBsToJpsiPhi:

```
from StrippingSelections import Bs2JpsiPhi
selSeq = Bs2JpsiPhi.sequence
Bs2JpsiPhiLine = StrippingLine('Bs2JpsiPhiLine'
                               , prescale = 1
                               , algos = [selSeq]
                               , stream = 'Bmuon'
                               )
```

See Juan's next talk.

```
from Configurables import DaVinci
from StrippingConf.Configuration import StrippingConf

StrippingConf().OutputType = "ETC"

DaVinci().EvtMax = 200
DaVinci().DataType = "2008"
DaVinci().Simulation = True
DaVinci().ETCFile = "etc.root"
```

This will load all selections from Phys/StrippingSelections and produce an ETC file called "etc.root" with selection results.

New OutputType = "SETC" added for tagger.

New DST writing syntax using Juan's MicroDST writer:

```
from Gaudi.Configuration import *
from Configurables import SelDSTWriter, StrippingConf
from Configurables import DaVinci

sc = StrippingConf()
sc.ActiveLines = [ ]
sc.ActiveStreams = [ ]
sc.OutputType = "DST"

dstWriter = SelDSTWriter("MyDSTWriter",
    SelectionSequences = sc.activeStreams(),
    OutputPrefix = 'Strip',
    OutputFileSuffix = '000000'
)

DaVinci().EvtMax = 1000
DaVinci().UserAlgorithms = [ dstWriter.sequence() ]
```

- `StrippingConf().MainOptions = "path/selections.py"`

Main options file to import.

- `StrippingConf().ActiveLines = ["B2HH", "B2Dpi"]`

List of active lines (take all if empty). Can be useful for debugging a specific selection.

- `StrippingConf().ActiveStreams = ["MuonPair", "DStar"]`

List of active streams (take all if empty).

- `StrippingConf().OutputType = "ETC"`

Output type: "ETC", "DST", "SETC", or "NONE"

- ETC writing not flexible enough (streams are appended to DaVinci inside `StrippingConf`). Want more transparent syntax as for DST writing.
- No possibility to modify streams content from `AppConfig` (needs releasing the patch version of `StrippingSelections`).
- Possibly, need different DST content (candidates/vertices/hits etc.) for each stream.
- Need structure similar to `HltDecReport` to know at what stage the event was rejected.

Proposal for a new StrippingSelections structure

- Move StrippingSelections to python/
- Stream definition in python/StreamBmuon.py

```
from StrippingConf.StrippingStream import StrippingStream
from StrippingSelections import StrippingBs2JpsiPhi
...

stream = StrippingStream("Bmuon")
stream.appendLines( [
    StrippingBs2JpsiPhi.line,
    ...
] )
```

Proposal for a new StrippingSelections structure

- Instead of importOptions with selections, do (e.g. ETC writing)

```
from StrippingConf.Configuration import StrippingConf
from StrippingSelections import StreamBmuon, ...
```

```
sc = StrippingConf()
sc.appendStream( StreamBmuon.stream )
sc.appendStream( ... )
```

```
from Configurables import EventTuple, TupleToolSelResults
```

```
tag = EventTuple("TagCreator")
tag.EvtColsProduce = True
tag.ToolList = ["TupleToolEventInfo", "TupleToolRecoStats", ...]
tag.addTool(TupleToolSelResults)
```

```
tag.TupleToolSelResults.Selections = sc.selections()
```

```
from Configurables import DaVinci
```

```
DaVinci().appendToMainSequence( [ sc.sequence() ] )
DaVinci().appendToMainSequence( [ tag ] )
DaVinci().ETCFile = "etc.root"
```

- StrippingConf simplified (just a list of StrippingStreams with functions to access sequencers, selection names etc. needed for DaVinci)
- Multiple instances of StrippingConf allowed (in case we need it to write DST with different content)
- Streams can be created and appended to StrippingConf in AppConfig (to redefine default ones).
- ActiveStreams and ActiveLines options will be left.
- But:
 - Easier to forget to include selection.
 - Syntax allows one selection to be included to many streams.
- Mostly implemented, but not released yet.