MicroDST / Stripping DST writing and reading

Juan Palacios

LHCb SW week

October 2009

Program for this session

- A few preliminaries about the MicroDST and Stripping DST
- Examples on how to write one
  - Will use the new Writer configurables
- Write one for the Bs selection we wrote
  - Show why it is easier with SelectionSequences
- Read it in GaudiPython
- Read it in C++

The MicroDST (1)

- The MicroDST is a small DST where you store only information you care about
- Already presented many times
- Typically for physics analysis:
  - Full particle decay trees including all intermediates and decay vertices
  - Particle -> PV relations
  - Re-fitted PV + particle -> re-fitted PV relations
  - Associated MCParticles
  - ProtoParticles, Tracks for particles in decay tree
  - Flavour tags
  - L0, Hlt decisions
- Very configurable and extendable
  - And advertised: https://twiki.cern.ch/twiki/bin/view/LHCb/MicroDST

The MicroDST (2)

- Two-fold data reduction (physics analysis context)
  - In a physics analysis, store only relevant information ***but*** only for events that we care about (typically those passing some selecion)
- Format recognised by our LHCb software tools
  - Store this information such that it can be analysed with DaVinci C++ algorithms, GaudiPython, etc.
  - Can act on stored data using standard LHCb physics tools
- Use Gaudi technology to get back to information on original DSTs if necessary
  - Meaning catalogues and SmartRefs

The Stripping DST

- A full DST with events that passed a stripping stream
  - Data reduction from event selection
  - Extra containers can be added
- Includes reduced MicroDST partition with candidates and full decay tree, plus PV relations
  - Contents of partition can be configured

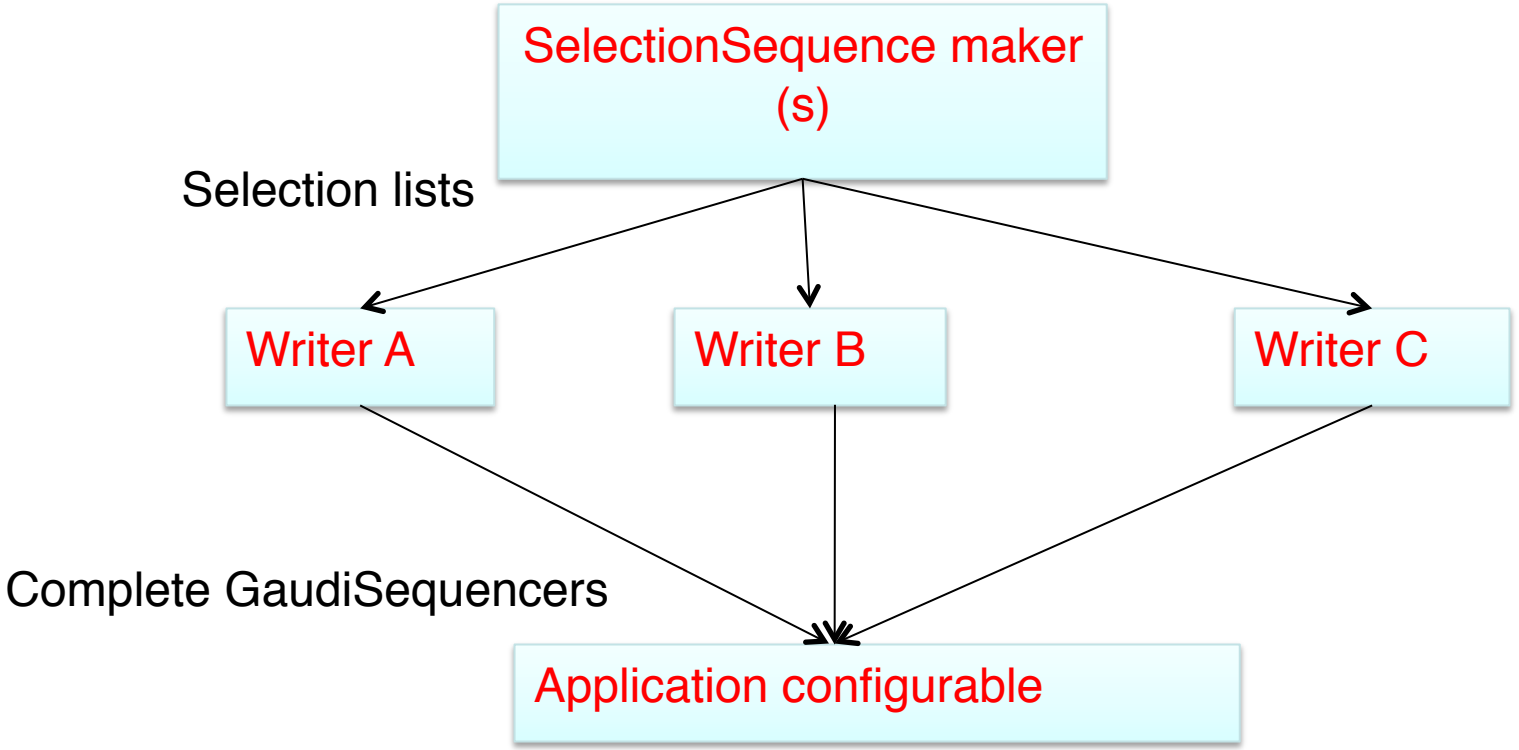
Stripping/Micro DST

- The concept of the MicroDST has been married to the stripping DST (DaVinci v24r2 onwards)
- They can be analysed in the same way
  - Bear in mind which information is available in each one
- They can be written with very similar semantics
  - DST writer idiom

The DSTWriter model (1)

1. The user creates some *SelectionSequences* (more later)
2. The DSTWriter Configurable that takes a list of *SelectionSequences* and creates a self-consistent sequencer that is made available to an application Configurable
   - No internal use of high level configurables (DaVinci, DstConfig, LHCbApp, DaVinciDSTWriter)
   - No reliance on anything that is not passed to it as a property
   - No modification of the inputs
   - No need to know what the selection sequences contain as long as they conform to a *certain interface*
3. A high level Configurable just takes the sequence and runs it
   - No need to know what the sequence contains
   - No need to set up any output streams, deal with DST names, etc.

The DST Writer model (2)



- Each layer is decoupled from the other
- The selection sequence hers is abstract. It could be many separate entities
- An application could write a MicroDST, a stripping DST and a selection DST from random combinations of selections (if that were desirable)

Hands-on writing

- Write a MicroDST with the Bs selection from the previous session, including
  - Particles
  - Associated MC particles
  - B-tags
- Write a stripping DST using the stripping line written in previous session

MicroDSTWriter example

```
from Configurables import DaVinci, MicroDSTWriter
importOptions("$STDOPTS/PreloadUnits.opts")
# import a SelectionSequence here, assume it is called MySelSeq
from SomeModule import SelSeq

dstWriter = MicroDSTWriter("MyDSTWriter",
                           SelectionSequences = [SelSeq],
                           CopyMCTruth = True,
                           CopyBTags = True,
                           CopyProtoParticles = True,
                           OutputFileSuffix = 'Test')


dv = DaVinci( UserAlgorithms = [dstWriter.sequence()], EvtMax = 5000,
              DataType = '2008')
dv.Input = [……]
```

That's it. Candidates will be written to '/Event/MicroDST/<some_name>'
You can get <some_name> from SelSeq.outputLocation()

DSTWriter for Stripping example

```
from Configurables import DaVinci, StrippingConf, SelDSTWriter

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

dstWriter = SelDSTWriter("MyDSTWriter",
                         SelectionSequences = sc.activeStreams(),
                         OutputFileSuffit = 'TutorialTest',
                         )


dv = DaVinci( UserAlgorithms = [dstWriter.sequence()], EvtMax = 5000,
              DataType = '2008')
dv.Input = [……]
```

That's it. Candidates will be written to '/Event/Strip/<StrippingLineName>'

Reading Micro/Stripping DSTs

- Differences with standard DSTs
  - Particles are in '/Event/Something/Phys'
    - Need to take care with InputLocations
  - Some information may not be there.
    - Do you or the framework assume it is?
- GaudiPython
  - Trivial. See Ex/MicroDSTExample/scripts
- C++ and some possible problems
  - For MicroDST, DAQ/ODIN and Rec/RecHeared locations may be assumed by framweork