Lancaster University

Experience Migrating ID algorithms to AthenaMT

Adam Barton Stewart Martin-Haugh, Ben Wynne

Introduction

- We have ported many of the InDet Algorithms and tools to produce an "InDetHive" example.
- Most of the time this is a simple process of moving the objects to DataHandle member variables and changing the record/retrieve code.
- The way datahandles interact with the python interface mean that any key string can be removed and you can pass the handle directly through declare property. This typically means the joboptions don't need to be changed.
- A typical Change log

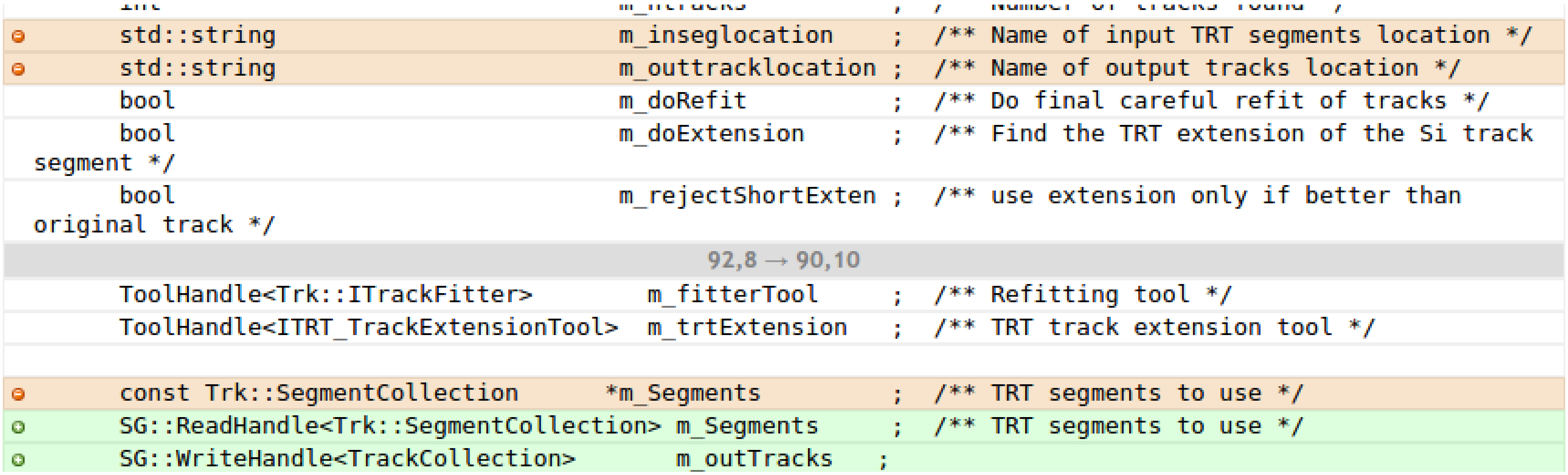
Lancaster University

Example of moving an alg to DataHandles – header file

Make sure any storegate pointer is a class member (if not already).

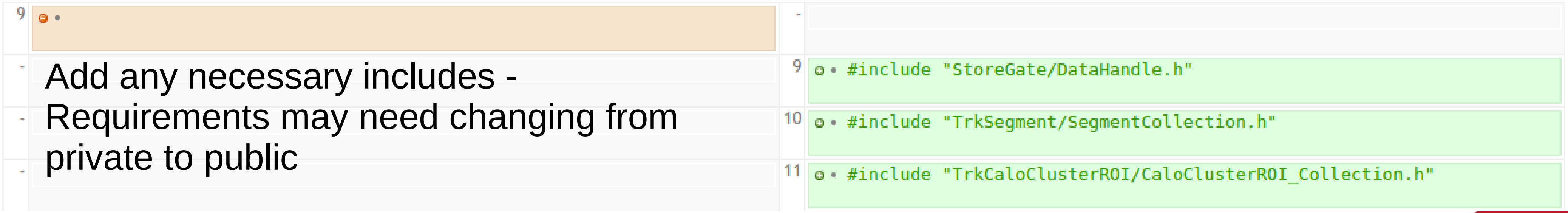
Wrap it in the appropriate template (WriteHandle or ReadHandle)

It is not necessary to have a string for the key as this is included in the template.



```
⊖       std::string                        m_inseglocation    ;  /** Name of input TRT segments location */
⊖       std::string                        m_outtracklocation ;  /** Name of output tracks location */
        bool                               m_doRefit          ;  /** Do final careful refit of tracks */
        bool                               m_doExtension      ;  /** Find the TRT extension of the Si track
  segment */
        bool                               m_rejectShortExten ;  /** use extension only if better than
  original track */
                                  92,8 → 90,10
        ToolHandle<Trk::ITrackFitter>        m_fitterTool       ;  /** Refitting tool */
        ToolHandle<ITRT_TrackExtensionTool>  m_trtExtension     ;  /** TRT track extension tool */

⊖       const Trk::SegmentCollection       *m_Segments         ;  /** TRT segments to use */
⊕       SG::ReadHandle<Trk::SegmentCollection> m_Segments      ;  /** TRT segments to use */
⊕       SG::WriteHandle<TrackCollection>         m_outTracks    ;
```

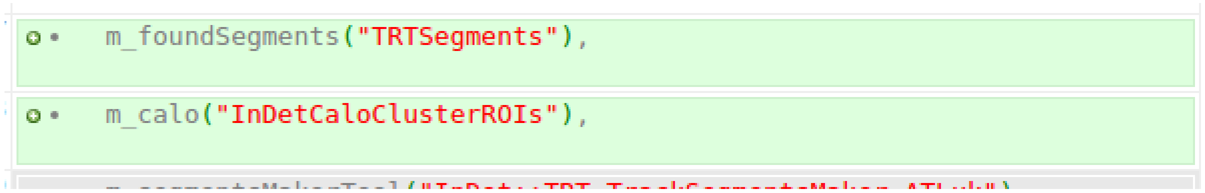


```
9  ⊖ •                                   -
-                                        9   ⊕ • #include "StoreGate/DataHandle.h"
-                                        10  ⊕ • #include "TrkSegment/SegmentCollection.h"
-                                        11  ⊕ • #include "TrkCaloClusterROI/CaloClusterROI_Collection.h"
```

Add any necessary includes - Requirements may need changing from private to public

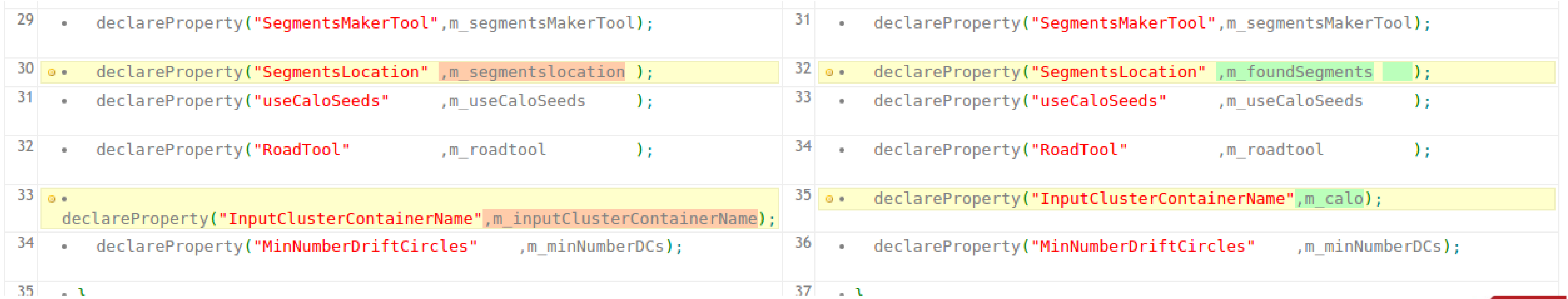
Example of moving an alg to DataHandles - constructor

- Initialise datahandles with default key string, in initialise list



```
m_foundSegments("TRTSegments"),
m_calo("InDetCaloClusterROIs"),
```

- Declare property can be used to pass the key string straight into the datahandle

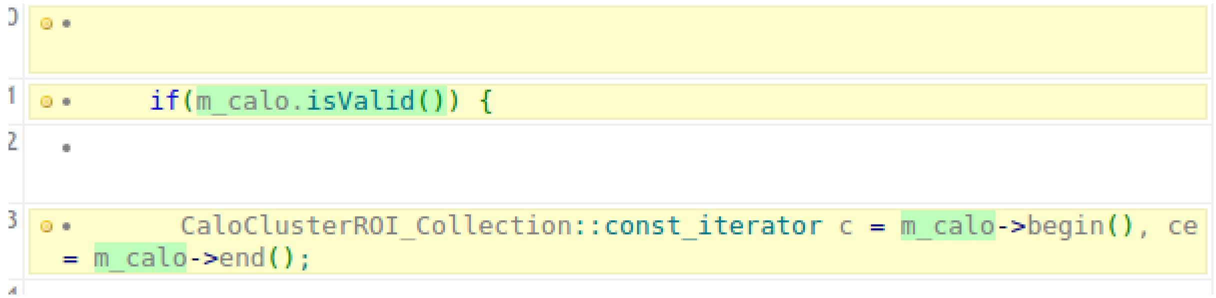


```
declareProperty("SegmentsMakerTool",m_segmentsMakerTool);
declareProperty("SegmentsLocation" ,m_segmentslocation );
declareProperty("useCaloSeeds"     ,m_useCaloSeeds     );
declareProperty("RoadTool"         ,m_roadtool         );
declareProperty("InputClusterContainerName",m_inputClusterContainerName);
declareProperty("MinNumberDriftCircles"    ,m_minNumberDCs);
}
```

```
declareProperty("SegmentsMakerTool",m_segmentsMakerTool);
declareProperty("SegmentsLocation" ,m_foundSegments   );
declareProperty("useCaloSeeds"     ,m_useCaloSeeds     );
declareProperty("RoadTool"         ,m_roadtool         );
declareProperty("InputClusterContainerName",m_calo);
declareProperty("MinNumberDriftCircles"    ,m_minNumberDCs);
}
```

ReadHandle

- To use a read handle simply check isValid(), then use as a normal pointer.



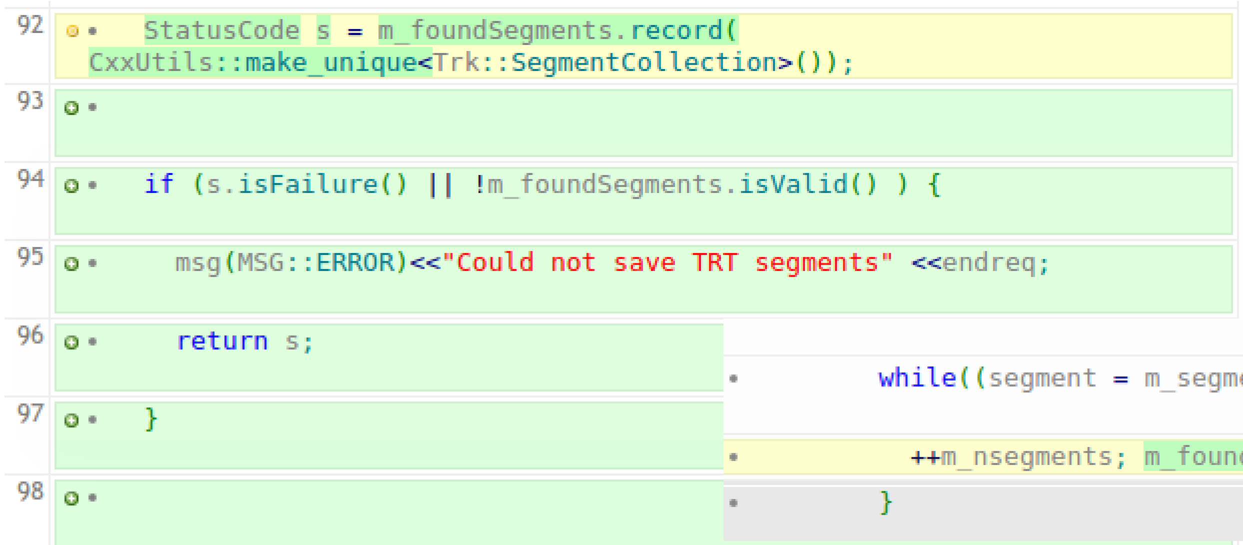
```
      if(m_calo.isValid()) {

        CaloClusterROI_Collection::const_iterator c = m_calo->begin(), ce
 = m_calo->end();
```

- If you need the simple pointer you can access it with .cptr(). NOTE: The handle retains ownership
- If you want to access the key (std::string) for the object call .name() - useful for debugging messages.

WriteHandles

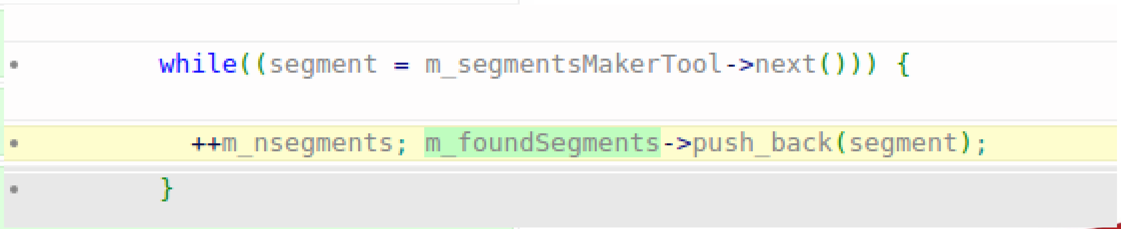
- Writehandles can own your created object and used throughout execute() to modify. The object will only be considered “complete” when execute() has finished.
- Tools that write objects will be considered complete when their calling algorithm has finished execute()
- If you have declared a (none blank) write handle you should use it, if only to create an empty container.



```
StatusCode s = m_foundSegments.record(
CxxUtils::make_unique<Trk::SegmentCollection>());

if (s.isFailure() || !m_foundSegments.isValid() ) {
  msg(MSG::ERROR)<<"Could not save TRT segments" <<endreq;
  return s;
}

```



```
while((segment = m_segmentsMakerTool->next())) {
  ++m_nsegments; m_foundSegments->push_back(segment);
}
```

Lancaster University

Crash: “Storegate Dump”

- ATR-13715 tracks a nasty bug that was causing crashes in athenaMT mode for some very unobvious reasons. Once isolated it was fixed in: AthenaServices-01-60-10

```
In AthenaHiveEventLoopMgr, clearStore() is called before the
EndEvent incident is fired. Since the event store dump debug
output is associated with EndEvent, this leads to all
DataProxies being reported as invalid, and worse, attempting
to make them valid again (the isValid() method can affect the
proxies, it's not const).
```

Issue: Optional Usage of Handles

- Many algorithms have functionality that is not used in every configuration mode. Objects are not needed or created in some situations. This leads to sub-optimal and buggy scheduling when the scheduler assumes every datahandle will be used.
- To solve this problem the scheduler has been changed to ignore handles with empty keys (empty at configuration time).
  - Job Options need to be modified to pass empty strings in the appropriate modes

Issue: Behaviour contigent on storegate object existance

- Some algorithms contain code that runs if a certain container exists in storegate.
- This should not exist in athenaMT, the scheduler must know what is required and the algorithm will only run once this is met.
- The conditional behaviour either needs to be removed or changed to a configuration conditional.
- May require consultation with the algorthim experts

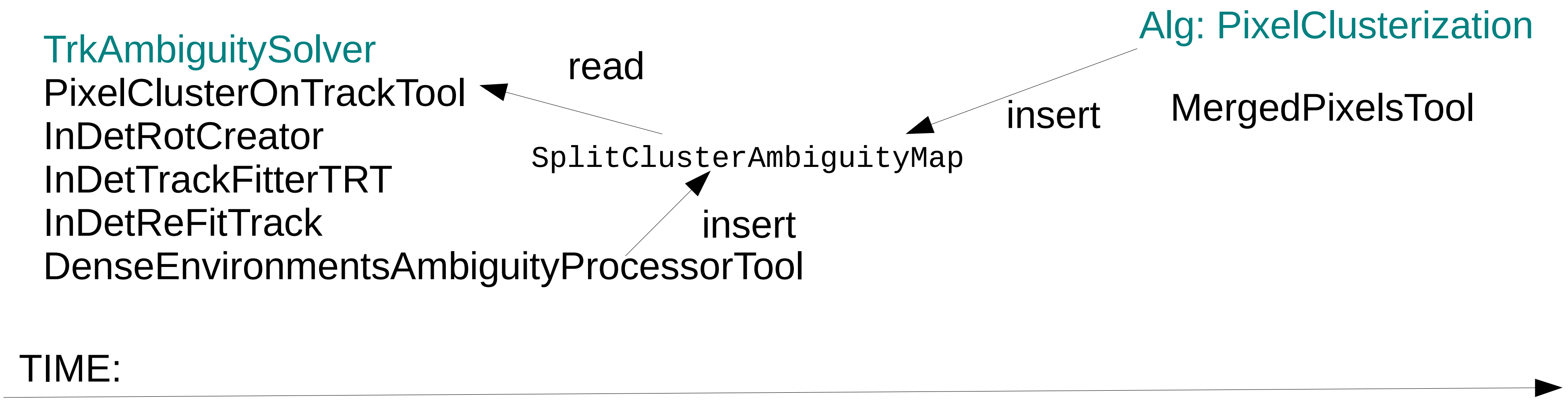
Issue: Tools with “Begin Event” Objects

- A common method we found is that tools will create collections using a “BeginEvent” incident and then add to the collection during calls.
- Problem: BeginEvent incidents may not be supported in athenaMT.
  - Remove the incident. Add a “newEvent” method to the tool and add this to the appropriate algorithms
- Public tools are no longer supported so if this tool requires public behaviour this will need to be changed.
  - We haven’t really found anything like this yet.

Issue: Nested tools passing data via common storegate object

- When the *DenseEnvironmentsAmbiguityProcessorTool* is used in the TrkAmbiguitySolver it creates a *SplitClusterAmbiguityMap* and inserts items into it. It invokes a chain of tools leading to the *PixelClusterOnTrackTool* which then reads from the map.
- This is problematic but not entirely forbidden:
  - The *PixelClusterOnTrackTool* can use the datahandle but mustn't declare it as a property. Doing so would mean the TrkAmbigiutySolver alg will appear to both create and require the same item (possibly leading to compatability issues with other algs).
  - If different threads are invoked during the tool chain we could have race conditions.
- This particular case isn't used in the trigger so I haven't implemented a solution.

Issue: Independant tools adding to a common object



- In current system the SplitClusterAmbiguityMap is created in a BeginEvent (or retrieved if the other alg already created it) and two algrothims can insert into the map.
- Assuming this behaviour is still required we must either change the maps to different keys and add a merging algorithm or implement a thread safe service to house the map.
- In such situations it would be a good idea to use the job options to collect the keys of collections that need merging and then pass this a merging algorithm.

Feature Request: Arrays of Keys

- It may be necessary for an alg to accept a (configuration time defined) number of keys of the same time, for say, merging.
- You can implement this already to a limited degree using optional handles or some hacks.
- We’ve asked for this to be supported fully as it may be very useful for certain tasks.

Recipe for current converted algs

- asetup 20.8.X-VAL,rel_3,here
- pkgco.py InDetRecExample-02-06-05
- cd InnerDetector/InDetExample/InDetRecExample
- patch -p0 < /afs/cern.ch/user/s/smh/public/InDetRecExample2.patch
- make -C cmt
- mkdir $TestArea/run
- cd $TestArea/run
- source RecExCommon_links.sh
- athena.py --threads=1 InDetRecExample/InDetHivePreExec.py RecExRecoTest/RecExRecoTest_RTT_id.py

Summary

- The majority of algorthims can be ported to datahandles transparently with no modifications needed for job options. (Some compile requirements may need changing).
- The most common issues requiring further changes are conditional handles needign to be given empty strings appropriately.
- Tools relying on BeginEvent and shared object behaviour need to be slightly reworked.
- Other "hacky" behavior exists and would need addresssing on a case by case basis.