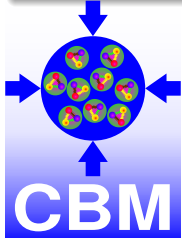


CBM online: updates in the code base



Sergei Zharko

CBM Department
GSI Helmholtzzentrum für Schwerionenforschung
Darmstadt, Germany

17.2.2025



- Goals
- Header inclusion
- Library management
- New reco-data classes and containers in online/offline

- **Purpose:** development of clean, transparent and efficient timeslice/event reconstruction algorithm
- **Goal:** replacement of “legacy” algorithms and data-classes with the new ones from `cbm::algo`
- **Goal:** only one instance of each data unpacking/reconstruction algorithm in `CbmRoot`
- **The algo code-base organization and full integration to the CbmRoot is to be carried out**



Case 2: Headers included with partial path

- Mostly in algo folder/libraries
- Flat install of headers not enforced there, mainly due to usage of namespaces

Cause:

- algo headers not consistently installed with full original path
 - Sometime flat directly in install folder
 - Sometime with partial path (e.g. "<install>/include/yaml" for a file from "<source>/algo/base/utis/yaml")
- Partial path used in include macro but not matching install path:
 - #include "<subfolder>/header.h"

⇒ Complex fix for single instance of problem: Need expert intervention

⇒ Long term need consistent convention, even if "it leads to long includes" or "it leads to long filenames"

- The solution with “long includes” was selected for headers in Algo/AlgoOffline

```
1 /// Example: algo/ca/TrackingSetup.h
2 #include "algo/base/SubChain.h"
3 #include "algo/detectors/sts/TrackingInterface.h"
4 #include "algo/detectors/tof/TrackingInterface.h"
5 #include "algo/detectors/trd/TrackingInterface.h"
6
7 /// Example: reco/steer/CbmSourceDigiTimeslice.h
8 #include "algo/base/DigiData.h"
9 #include "algo/global/RecoResultsInputArchive.h"
10 #include <FairSource.h>
```

- Headers are installed in nested directories starting from algo:

```
1 install(DIRECTORY base DESTINATION include/algo PATTERN "*.h")
2 install(DIRECTORY ca
3         DESTINATION include/algo
4         FILES_MATCHING
5         PATTERN "ca/core" EXCLUDE # NOTE: installed from another library
6         PATTERN "*.h")
```

- Merge request [!2084](#)

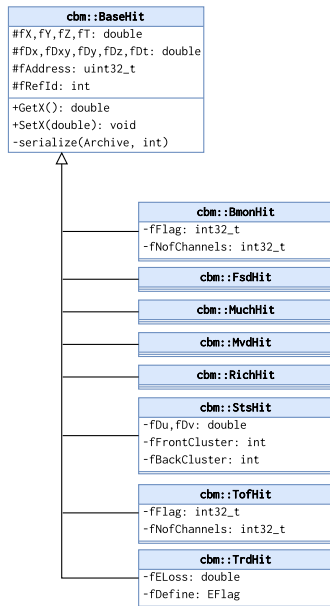
- Issues:
 - two versions of libraries: XXX and XXXOffline: handling of different loggers and NO_ROOT derictive
 - Algo/AlgoOffline contains useful features, which are generic and should not be a part of it
 - the features are reused in many places, which brings unnecessary dependency on Algo/AlgoOffline
- Migration of the features to separate libraries: CbmUtility and CbmContainers/OnlineContainers
- CbmUtility (merge request !2086):
 - extension to the C++ STL and BOOST: utilities to work with templates, enumerations, generic algorithms
 - no detector/analysis specific classes, no ROOT dependencies
 - namespace cbm::util
 - first class cbm::util::EnumDict
 - other related classes will be moved soon (**review of classes is needed**)
- CbmContainers/OnlineContainers (merge request !2085)
 - extension to the C++ STL and BOOST: different containers
 - no detector/analysis specific classes, no ROOT dependencies
 - namespace cbm
 - first classes: cbm::PODVector, cbm::PartitionedVector, cbm::PartitionedSpan
 - other related classes will be moved soon

New reconstructed data structures

- Purpose: online-friendly uniform data structures for reconstruction algorithms
- Self-contained representation of reconstructed event/timeslice
- Remastered Hit and Track classes
- Merge request [!1983](#) [WIP]

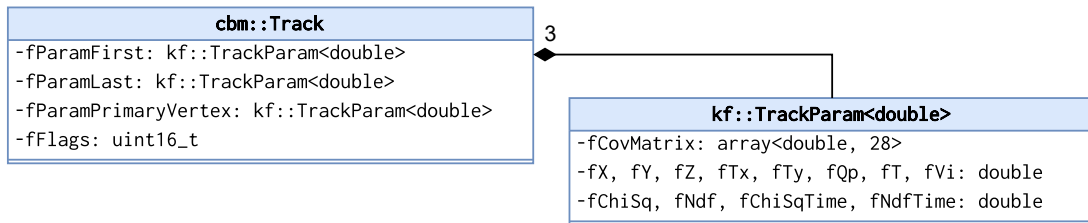
New reconstructed data structures: hit classes

- Base class – a pixel hit (x, y, z, t)
- No custom detector layout classes allowed, only CBM 32-bit address
- Derived classes add extra fields, **but do not override getters of base class**
- Derived classes are accessible via `template<ECbmModuleId> cbm::Hit_t:`
 - e.g.: `cbm::Hit_t<ECbmModuleId::kSts>`, `cbm::Hit_t<ECbmModuleId::kTof>`
 - easily accessible in template functions with the `ECbmModuleId` template parameter



Track class

- Track parameters in three points: first hit, last hit and primary vertex
- Track parameter class – from KfCore, naturally used by CA Tracking
- No information on individual hits (only flags)



New reconstructed data structures: RecoData class

cbm::RecoData	
-fHitContainer:	HitContainer
-fGlobalTrackContainer:	GlobalTrackContainer_t
-fDataType:	EDataType
+GetGlobalTrackContainer():	const GlobalTrackContainer_t&
+GetGlobalTracks():	span<const Track>
+GetGlobalTrack(int iTrk):	const Track&
+GetGlobalTrackHitIds(int iTrk):	span<const HitId>
+GetGlobalTrackHitIdsIn<ECbmModuleId>(int iTrk):	span<const HitId>
+GetHitIn<ECbmModuleId Id>(int iHit):	const Hit_t<Id>&
+GetHit(ECbmModuleId id, int iHit):	const BaseHit&
+GetHit(HitId):	const BaseHit&
+GetHitsIn<ECbmModuleId Id>():	span<const Hit_t<Id>>
+GetHitContainer():	const HitContainer&
+GetPartitionOfHitsIn<ECbmModuleId Id>(int iP):	span<const Hit_t<Id>>
+GetNoHitsIn<ECbmModuleId Id>():	int
+GetNoGlobalTracks():	int
+HasHits(ECbmModuleId):	bool
+HasGlobalTracks():	bool

cbm::TrackContainer		class Quality
-fTrackHitMatchContainer:	TrackHitMatchContainer	
-fvTracks:	vector<Tracks>	
-fvTrackQuality:	vector<Quality>	
+GetTrackHitMatchContainer():	const TrackHitMatchContainer&	
+GetHitIds(int iTrk):	span<const HitId>	
+GetHitIds(int iTrk, ECbmModuleId):	span<const HitId>	
+GetHitIdsIn<ECbmModuleId>(int iTrk):	span<const HitId>	
+GetNoHitsIn<ECbmModuleId>(int iTrk):	int	
+GetNoHits(int iTrk):	int	
+GetNoHits(int iTrk, ECbmModuleId):	int	
+GetQuality(int iTrk):	const Quality&	
+GetTrack(int iTrk) const:	const Track&	
+GetQuality(int iTrk):	const Quality&	
+SetQuality(const Quality&):	void	

cbm::HitContainer	
-fvHitsBmon:	PartitionedVector<BmonHit>
-fvHitsMvd:	PartitionedVector<MvdHit>
-fvHitsSts:	PartitionedVector<StsHit>
-fvHitsRich:	PartitionedVector<RichHit>
-fvHitsMuch:	PartitionedVector<MuchHit>
-fvHitsTrd:	PartitionedVector<TrdHit>
-fvHitsTof:	PartitionedVector<TofHit>
-fvHitsFsd:	PartitionedVector<FsdHit>
+GetHit(ECbmModuleId, int):	const BaseHit&
+GetHit(const HitId&):	const BaseHit&
+GetHitIn<ECbmModuleId Id>(int):	const Hit_t<Id>&
+GetHitIn<ECbmModuleId Id>(int iP, int iHit):	const Hit_t<Id>&
+GetHitsIn<ECbmModuleId Id>():	span<const Hit_t<Id>>
+GetPartitionOfHitsIn<ECbmModuleId Id>(int iP):	span<const Hit_t<Id>>
+GetNoHitsIn<ECbmModuleId Id>():	int
+GetNoHits(ECbmModuleId):	int
+GetHitsVectorIn<ECbmModuleId>() const:	PartitionedSpan<const Hit_t>
+GetHitsVectorIn<ECbmModuleId>():	PartitionedVector<Hit_t>&

cbm::TrackHitMatchContainer	
-fvHitIds:	vector<HitId>
-fvTrackHitMatches:	vector<TrackHitMatch>
+GetHitIds(iTrk):	span<const HitId>
+GetHitIds(iTrk, ECbmModuleId):	span<const HitId>
+GetHitIdsIn<ECbmModuleId>(iTrk):	span<const HitId>
+GetNoHits(iTrk):	int
+GetNoHits(iTrk, ECbmModuleId):	int
+GetNoHitsIn<ECbmModuleId>(iTrk):	int

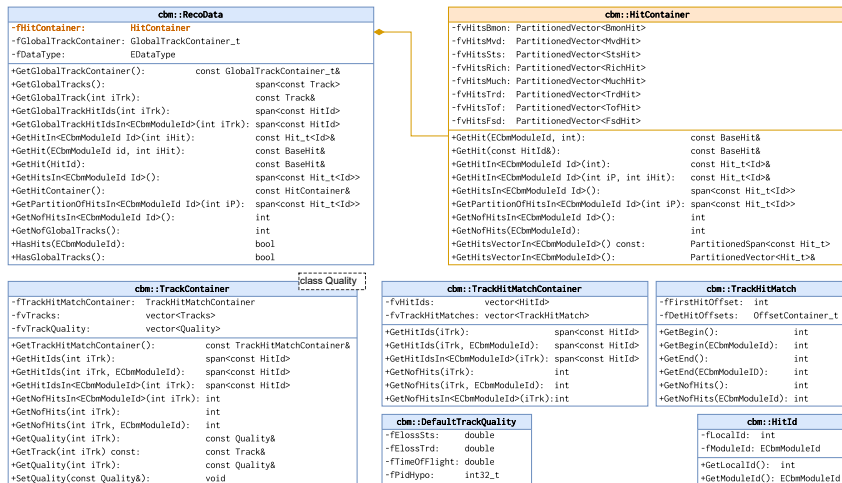
cbm::DefaultTrackQuality	
-fElossSts:	double
-fElossTrd:	double
-fTimeOfFlight:	double
-fPidHypo:	int32_t

cbm::TrackHitMatch	
-fFirstHitOffset:	int
-fDetHitOffsets:	OffsetContainer_t
+GetBegin():	int
+GetBegin(ECbmModuleId):	int
+GetEnd():	int
+GetEnd(ECbmModuleId):	int
+GetNoHits():	int
+GetNoHits(ECbmModuleId):	int

cbm::HitId	
-fLocalId:	int
-fModuleId:	ECbmModuleId
+GetLocalId():	int
+GetModuleId():	ECbmModuleId

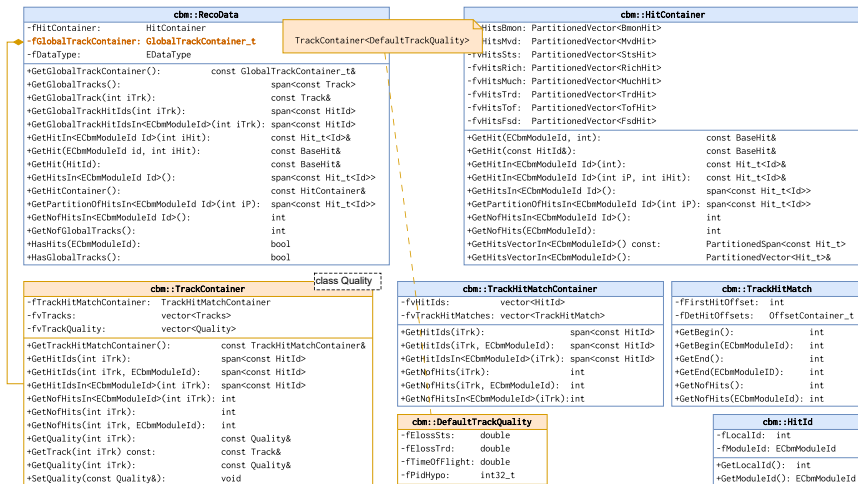
- Self-contained output format for timeslice/event reconstruction in CBM
- Contains tracks, hits, matching information, PID

New reconstructed data structures: HitContainer class



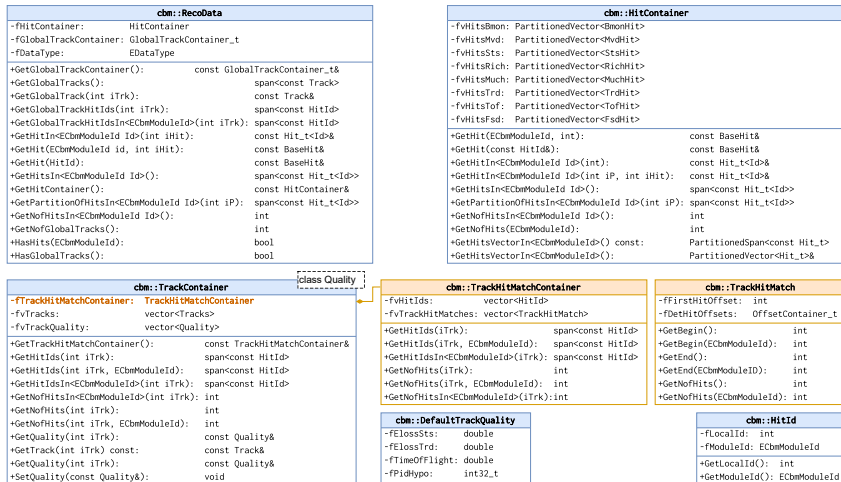
- Vectors of hits in different detectors
- Uniform access with run-time and compile-time detector ID selection

New reconstructed data structures: TrackContainer class



- Vectors of tracks, track quality (PID information), and track-hit matching

New reconstructed data structures: TrackHitMatchContainer class



- A flat vector of IDs of hits attached to tracks and vector of track-hit matches
- Track-hit match: a light-weight object, contains an index of first hit for a track in IDs vector and hit number offsets from different detector subsystems

- The new data classes are delivered, integrated to the online and being tested
- Next step – implementation of common reconstruction class (digi-timeslice/digi-event in online binary/offline task)