A Quality Assurance Task For Time-Based Track Reconstruction

Jenny Regina

Uppsala University
Department of Physics and Astronomy

PANDA Collaboration Meeting, GSI
November 4-8, 2019
Computing Session

Outline:

1. **Tracking Quality Assurance**
2. **Time-based data storage in ROOT**
3. **Time-based tracking Quality Assurance**
4. **Issue**

What is tracking Quality Assurance?

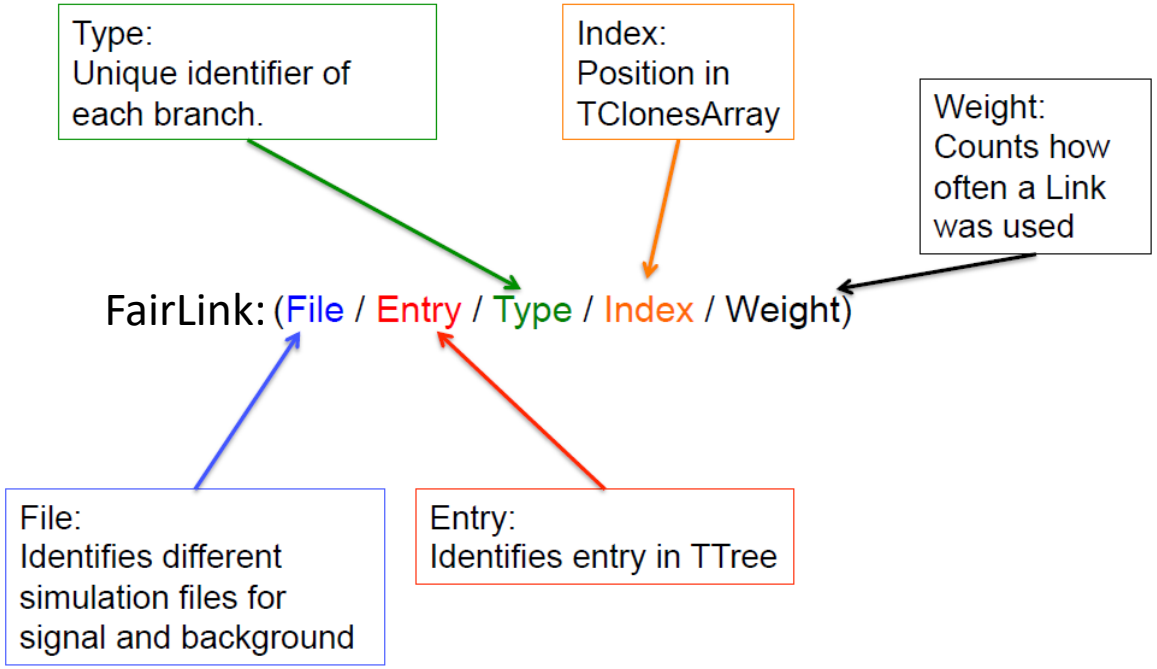
- **Q**uality **A**ssurance (QA) for tracking algorithms
- A track can be:
  - *Fully found* (all hits found)
  - *Partially found* (majority of found hits belong to the same track but contamination is possible)
  - *Spurious found* (>70% of all hits belong to one track, contamination not possible)
  - *Clones* (one track was found more than once)
  - *Ghosts* (reco track does not correspond to a MC track)
- Momentum resolutions
- Number of *true*, *false* and *missing* hits / track

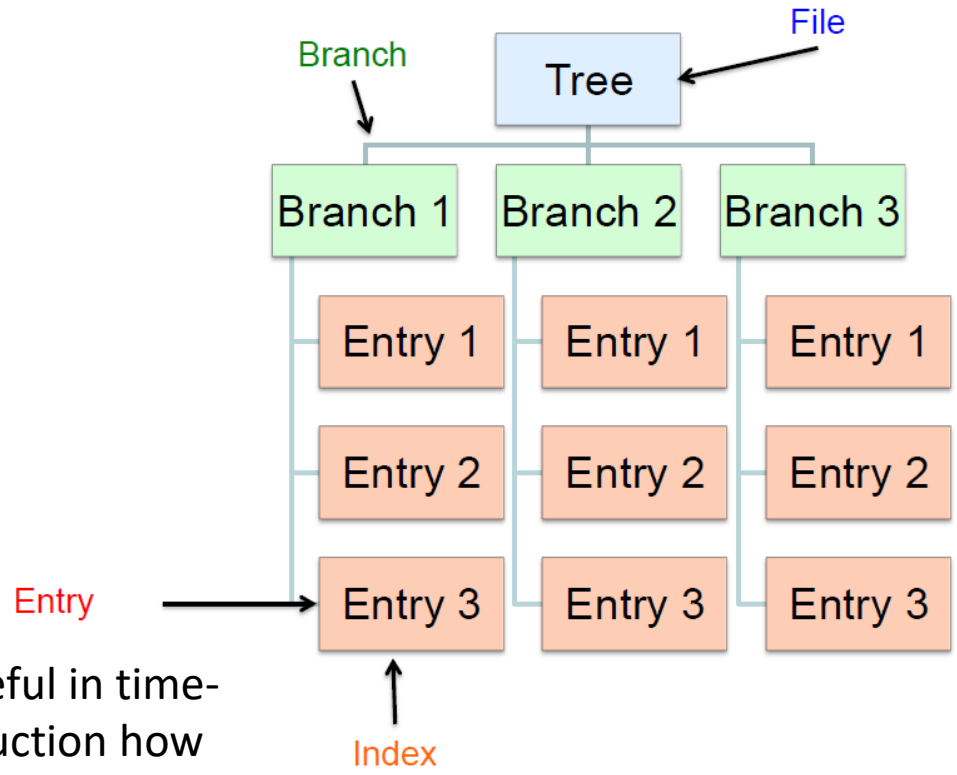
What is tracking Quality Assurance?

- To give “fair” representation of track finding algorithm: compare to only tracks which could be reconstructed in relevant detectors
- Use IdealTrackFinder with certain functor [1]
  - *E.g.* only save tracks with more than 5 STT hits, 6 STT+MVD+GEM hits …

[1] Lets you create objects which look like function

Event Based vs. Time Based Data Storage





Event-based data storage:

Branch: data objects, *eg.* Hits, tracks

Entry: event number

Index: Position in TClonesArray, in event-based simulation same for MC tracks and IdealTracks

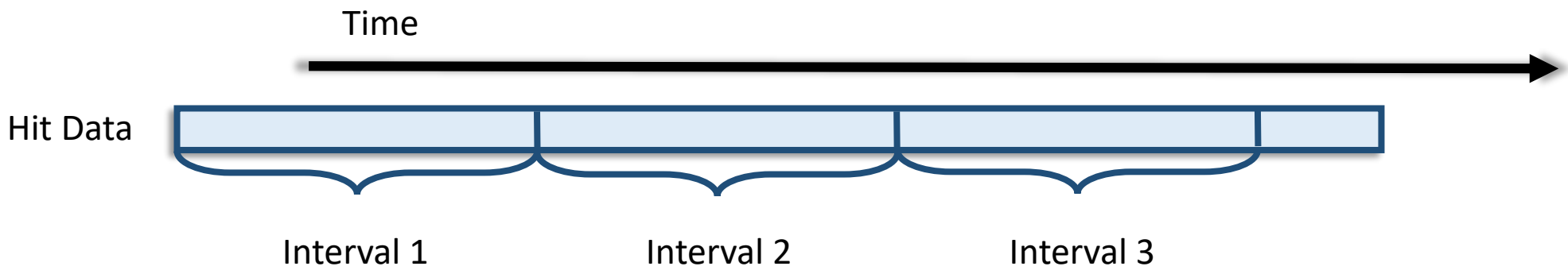
Need to be careful in time-based reconstruction how this is done!

Not the case for time based reconstruction!

Fetching the Time-Based Data

- Overwrite *Exec()* function with *GetData()* fetching certain data
- *E.g.* StopTimeFunctor
  - Fetches data from within a certain time-interval
  - Can only fetch data once! Skips data already collected

FairRootManager::Instance() →GetData(fSTTHitBranch, fFunctor, fStopTimeValue)



Simulation approach

- FairRoot 18.2.0
- FairSoft jun19
- PandaRoot Development Branch
- PndSttHitProducerRealFull

Running Time-Based Digitization

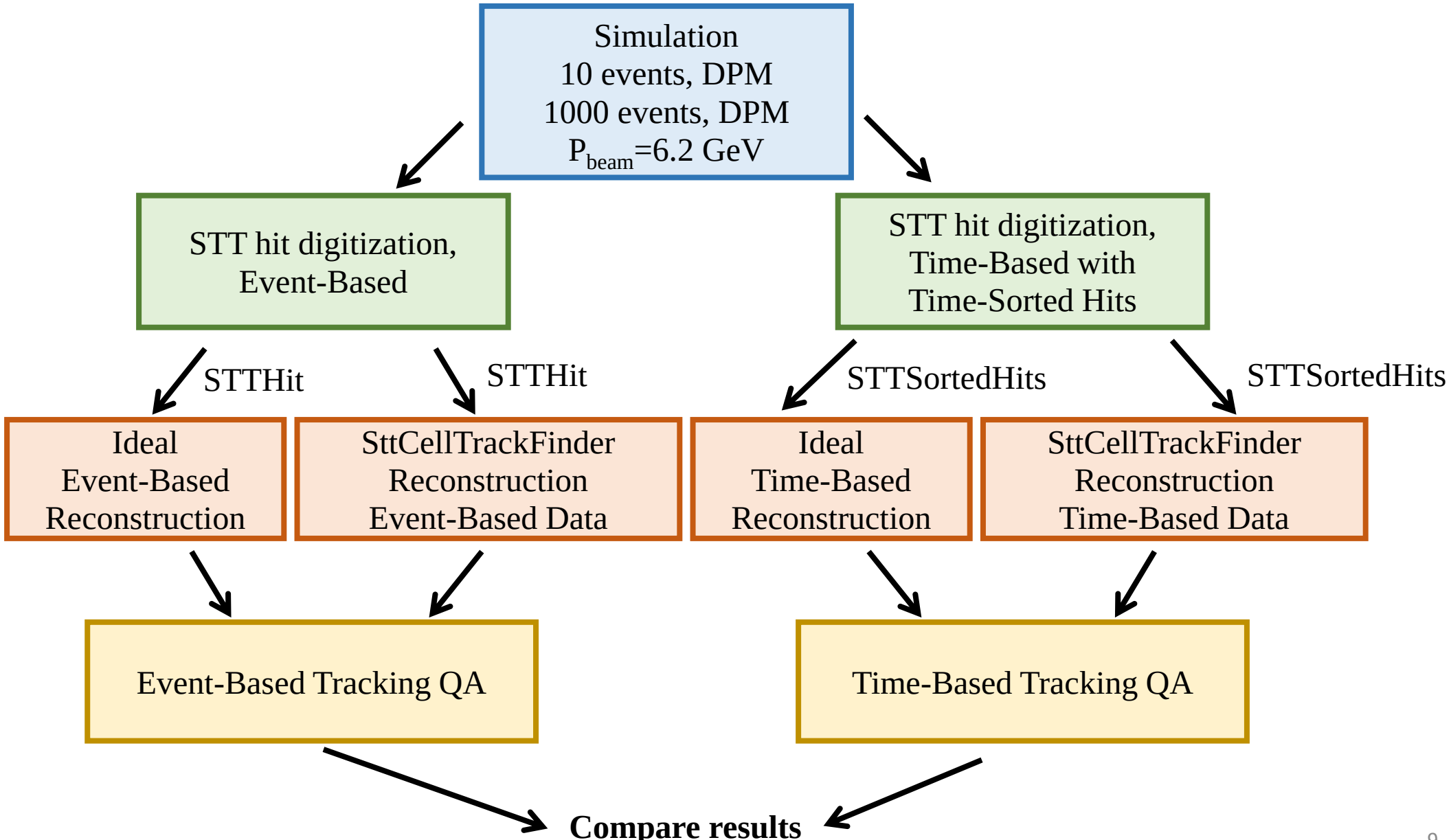
```
FairRunAna *fRun= new FairRunAna();
FairFileSource *fFileSource = new FairFileSource(inFile);
//fFileSource->ReadEventTimeFromFile("EventTimes.dat");
//fFileSource->SetEventMeanTime(50);
fFileSource->SetEventTimeInterval(500,501); // Time between events [ns]
fFileSource->SetBeamTime(1600, 400);
fRun->SetSource(fFileSource);
```

Note: **PndSttHitProducerRealFast** does not produce time-stamps

```
PndSttHitProducerRealFull* sttHitProducer = new PndSttHitProducerRealFull();
sttHitProducer->RunTimeBased();
fRun->AddTask(sttHitProducer);

PndSttHitSorterTask* sttSorter = new PndSttHitSorterTask(5000, 50, "STTHit", "STTSortedHits", "PndSTT");
fRun->AddTask(sttSorter);
```

Running the timeBasedTrackingQA

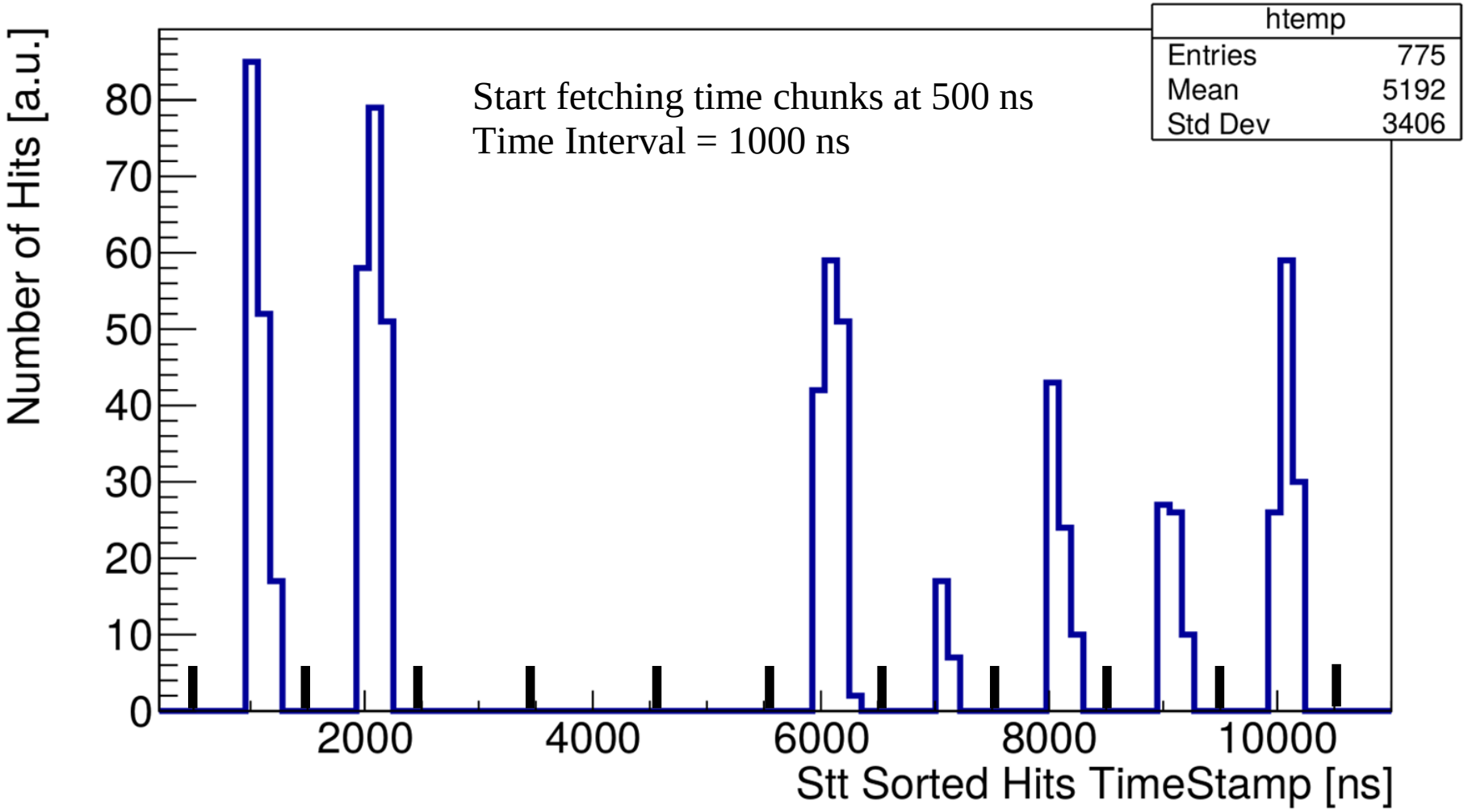
```
PndIdealTrackFinder* trackIdeal = new PndIdealTrackFinder();
trackIdeal->SetRelativeMomentumSmearing(0.05);
trackIdeal->SetVertexSmearing(0.05, 0.05, 0.05);
trackIdeal->SetTrackingEfficiency(1.);
trackIdeal->SetTrackSelector("OnlySttTimeBasedFunctor"); // Default StandardTrackFunctor
trackIdeal->AddBranchName("STTSortedHits"); // Default MVDHitsPixel, MVDHitsStrip, STTHit, GEMHit, FTSHit
trackIdeal->SetRunTimeBased(kTRUE); // Default kFALSE
fRun->AddTask(trackIdeal);

PndSttCellTrackFinderTask* cellTrack = new PndSttCellTrackFinderTask();
cellTrack->AddHitBranch("STTSortedHits");
cellTrack->SetRunTimeBased(kFALSE); // Default kFALSE
cellTrack->SetRunWithSortedHits(kTRUE); // Default kFALSE
fRun->AddTask(cellTrack);

PndTrackingQATask* trackingQA = new PndTrackingQATask("CombiTrack", "IdealTrack");
trackingQA->SetFunctorName("OnlySttTimeBasedFunctor"); // Default StandardTrackFunctor
trackingQA->AddHitsBranchName("STTSortedHits"); // Default MVDHitsPixel, MVDHitsStrip, STTHit, GEMHit, FTSHit
trackingQA->SetRunTimeBased(kTRUE); // Default kFALSE
trackingQA->SetVerbose(0); // Deafult 1 (derived from FairTask)
fRun->AddTask(trackingQA);
```


Simulation
10 events, DPM
1000 events, DPM
P_beam=6.2 GeV
STT hit digitization, Event-Based
STT hit digitization, Time-Based with Time-Sorted Hits
STTHit
STTHit
STTSortedHits
STTSortedHits
Ideal Event-Based Reconstruction
SttCellTrackFinder Reconstruction Event-Based Data
Ideal Time-Based Reconstruction
SttCellTrackFinder Reconstruction Time-Based Data
Event-Based Tracking QA
Time-Based Tracking QA
Compare results

Time Distribution of Hits

1. Events well separated in time
2. Possible to cut all data **in between** events

Should give results similar to event based case



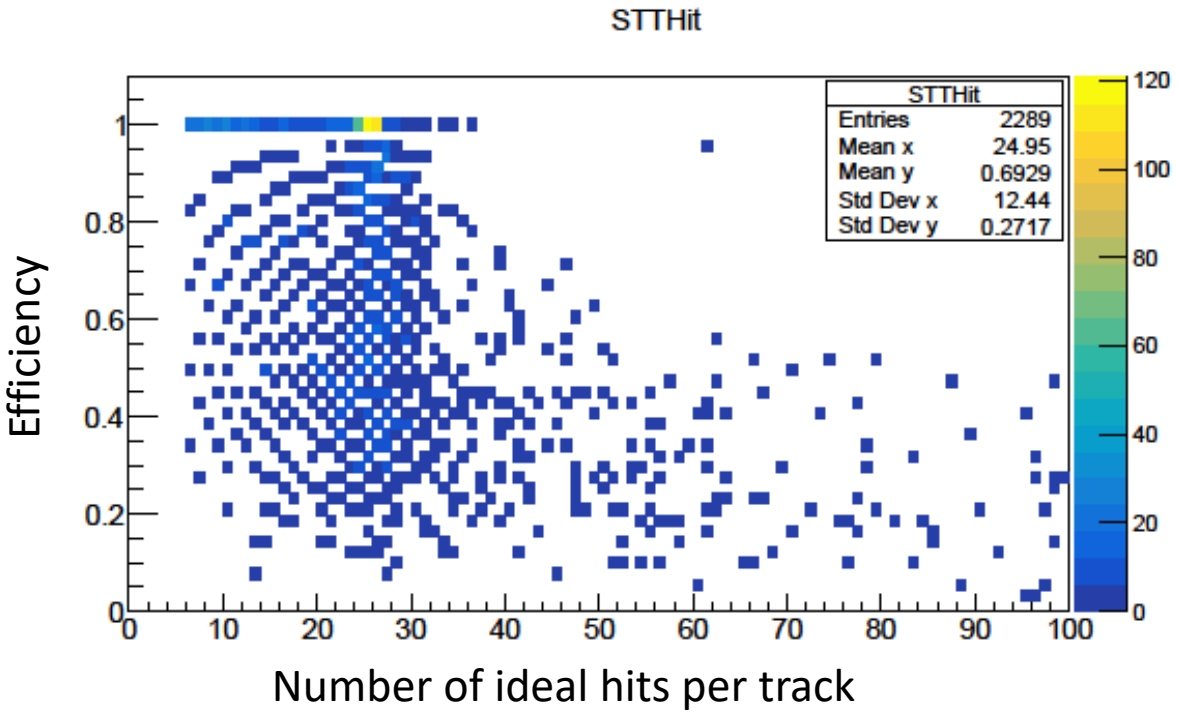
Results

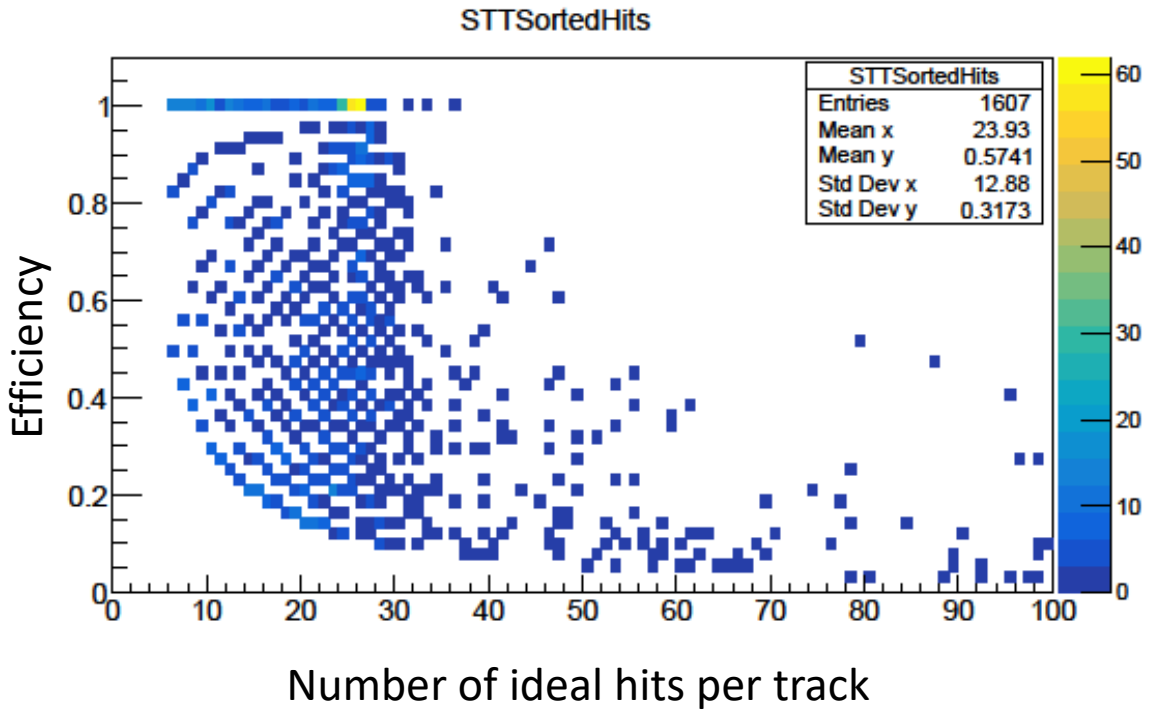
	Event based	Time based
All possible tracks	25	25
Primary tracks possible	9	9
Not found	0	0
Secondary tracks possible	16	16
Not found	1	1
Fully found	5	5
Partly found	17	17
Spurious	2	2
Ghosts	1	1
Clones	16	13

Agrees!

Counting performed in same way, difference might be due to inability to use GetCloneOfLinkData() function when running time based

Efficiencies





- Histograms somewhat difficult to interpret since events start getting cut
- Efficiency plot somewhat distorted in time based case
- Efficiency goes down faster with increasing number of hits for time based case

Issue

#195 GetCloneOfLinkData function not working in PndTrackingQA.cxx

Might be caused by wrongly set FairLinks in time based reconstruction

Workaround example:

1. Get FairLink of of Reco track
2. Get FairLink of MC track from map between the FairLinks
3. Compare MC track FairLink to all FairLinks in MC track TclonesArray
4. Get MC track object with the same FairLink

Danger with workaround: all objects might not be present in the TClonesArray as fetched by the *Exec()* function in current iteration – Might miss objects and get too low efficiency

Time-based specific features

Event purity: $\frac{n_{hits,\,correct\ event}}{n_{hits,total}}$

$n_{hits,\,correct\ event}$ - Number of hits with the same entry as the Reco track

$n_{hits,total}$ - Number of hits in track

Implemented - feature not giving output in Dev branch yet

***Clone Analyzer*:** testing how many hits are shared between original track and its clones

***Original track*:** the track with the most hits linked to the MC track
***Clone track*:** the rest of the tracks

Not implemented yet – focus has been put on testing and fixing time-based option

Summary

- The PANDA Quality Assurance task for tracking (PndTrackingQA) has been adapted to work for time-based reconstruction
- Two modes available, event-based (fast, MC-truth matching with indices) and time-based (slow, MC-truth matching with FairLinks)
- IdealTrackFinder, BarrelTrackFinder and SttCellTrackFinder can run on time-sorted hits

Outlook

- Find and fix issue causing GetCloneOfLinkData(FairLink) to fail in PndTrackingQA.cxx
- Merge additional features into Dev

Summary

- The PANDA Quality Assurance task for tracking (PndTrackingQA) has been adapted to work for time-based reconstruction
- Two modes available, event-based (fast, MC-truth matching with indices) and time-based (slow, MC-truth matching with FairLinks)
- IdealTrackFinder, BarrelTrackFinder and SttCellTrackFinder can run on time-sorted hits

Thank You!

Backup

Definitions

1. MC track
   - simulated track at simulation stage
2. Ideal track
   - track reconstructed by IdealTrackTinder
3. Reco track
   - track reconstructed by some realistic track finder (In this case the SttCellTrackFinder)

Algorithm	Accepts time sorted hits	If not: can be adjusted to accept time sorted hits (efforts required)	Algorithm takes time information into consideration	If not: can be adjusted to take time information into consideration (efforts required)
Ideal Track Finder	Yes	--	--	--
Barrel Track Finder	Yes	--	No	Needs closer look
Central Tracking	No	Relatively little effort, could probably be done in similar way as for the above algorithms	No	Needs closer look but one might be able to use time information to cluster hits
SttCellTrackFinder	Yes	--	Yes	--
CA Tracker	No	Little effort, can be done similarly as for above algorithms	No	Nees closer look but if needed could probably be done similarly as for SttCellTrackFinder
Pattern matcher	No	Little effort but might not be desirable at the moment	No	Patterns consist of tube and sector ids. Quite some effort is needed to create patterns with time information groups of hits can be matched to
Tracking QA	Yes	--	--	--
Unassigned hits task	No	Little effort, already assumes user given branch names	--	--

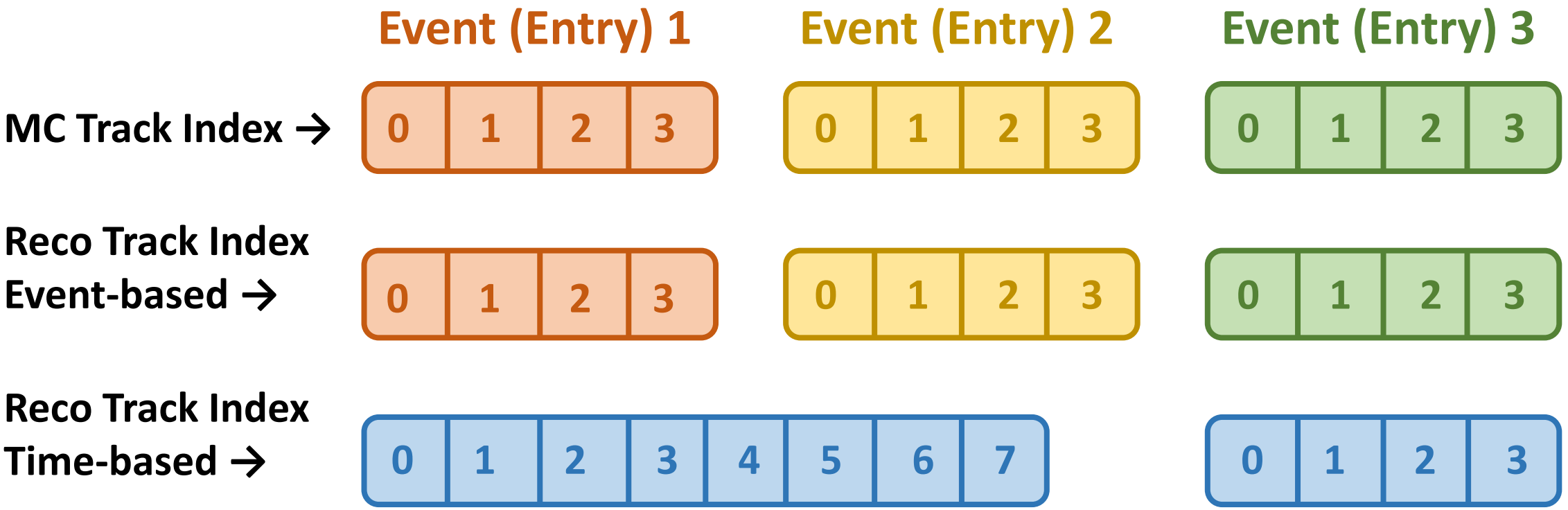
Performance

- std::chrono C++ library
- 1000 DPM events
  - EventTimeInterval = (1000,1001) in time based digitization
  - fStopTime=1000 ns
  - Start fetching data at 500 ns
- *Init()* of PndTrackingQA class is called in *Exec()* of *PndTrackingQATask.cxx*

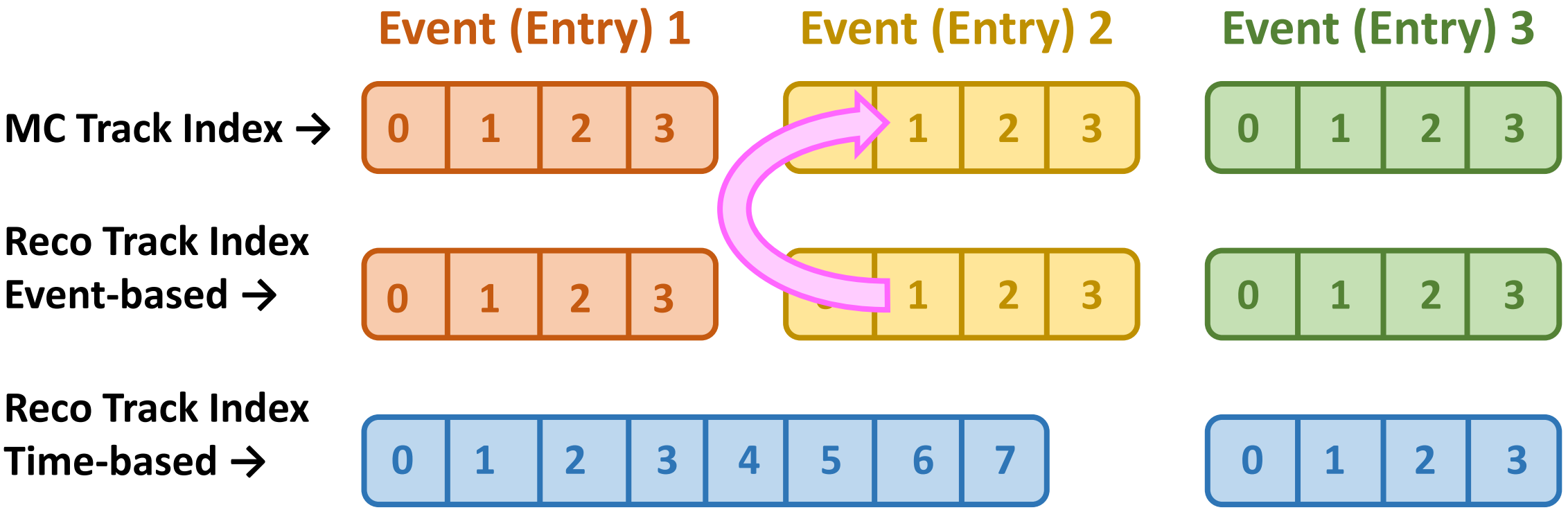
Mode	**Total time [s]**	**Mean Time/ Event [s]**	**Total time [s]**	**Mean Time/ Event [s]**
	Excluding Init() of PndTrackingQA		**Including** Init() of PndTrackingQA	
Event based	1.4	0.001	1.4	0.001
Time based	521.9	0.5	518.7	0.5

- Event based mode roughly 500 times faster than Time based mode
- However: right now a lot of loops and workarounds in time based mode
- Additional time due to *Init()* of *PndTrackingQA.cxx* seems to be within timing uncertainties

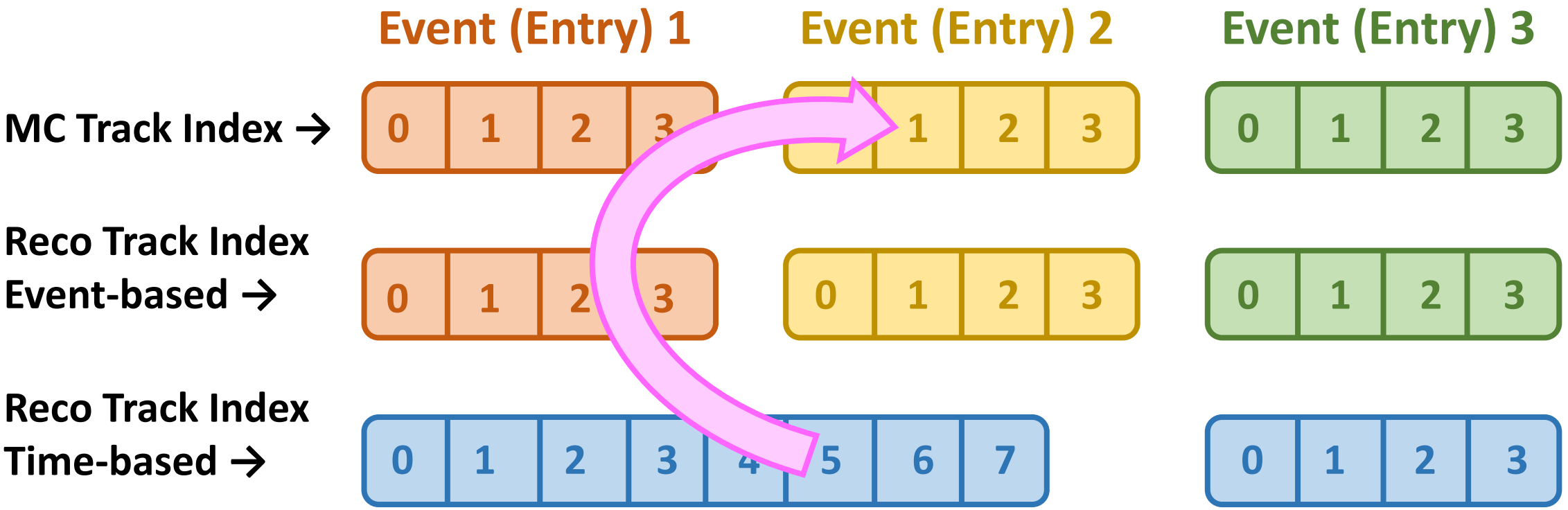
Event Based vs. Time Based Data Storage



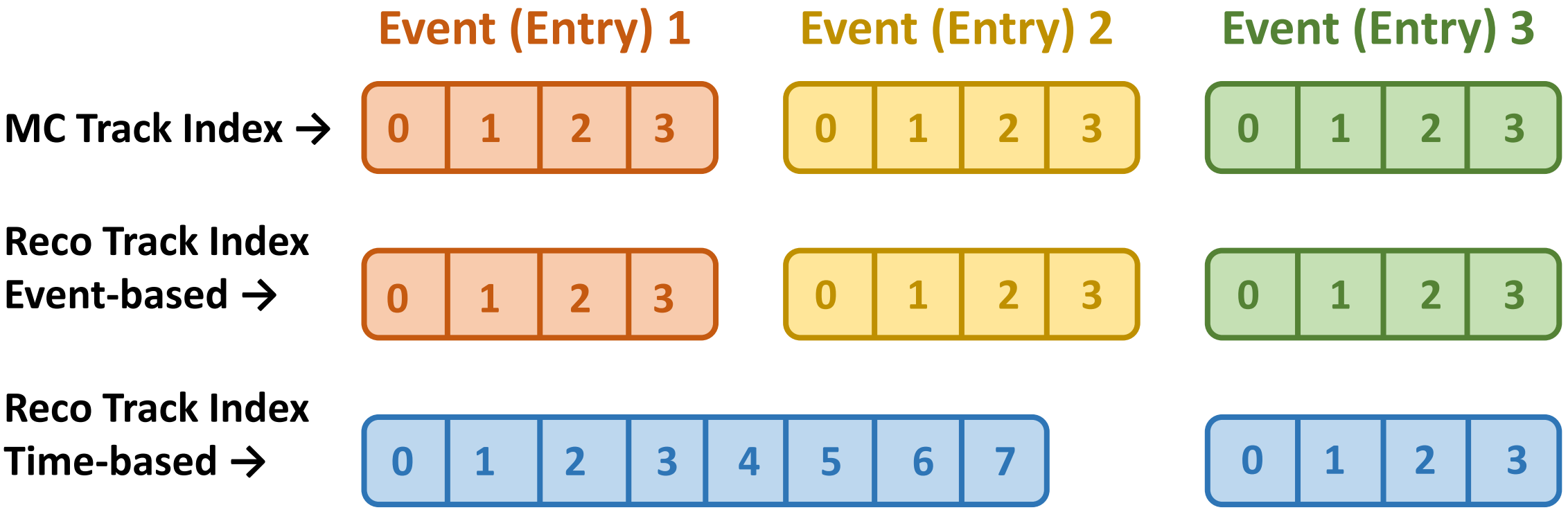
Event Based vs. Time Based Data Storage



Event Based vs. Time Based Data Storage



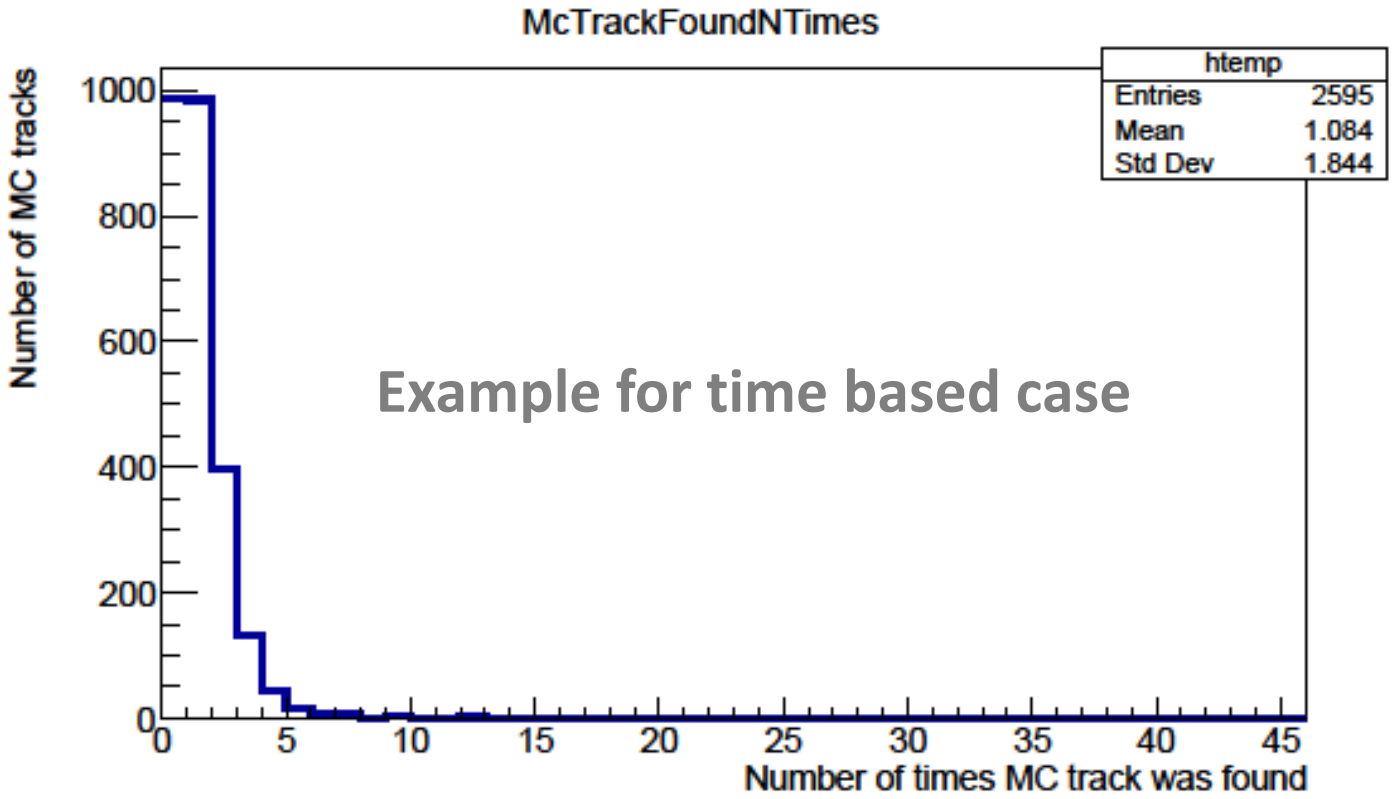
Event Based vs. Time Based Data Storage



Event-based: possible to do Monte-Carlo truth matching with indices
Time-based: must use FairLinks between data objects to do Monte-Carlo truth matching

Clones per track

QA root file -> qaTuple -> McTrackFoundNTimes



Bad idea to look at <u>only</u> number of clones!
If a tracking algorithm has *e.g.* 1000 clones you don't know if one track was found an additional 1000 times or if 1000 tracks were found twice

- Here one can see if majority of clones is *e.g.* made from very few tracks
- In this case ~400 tracks are found twice and < 200 tracks are found more than twice