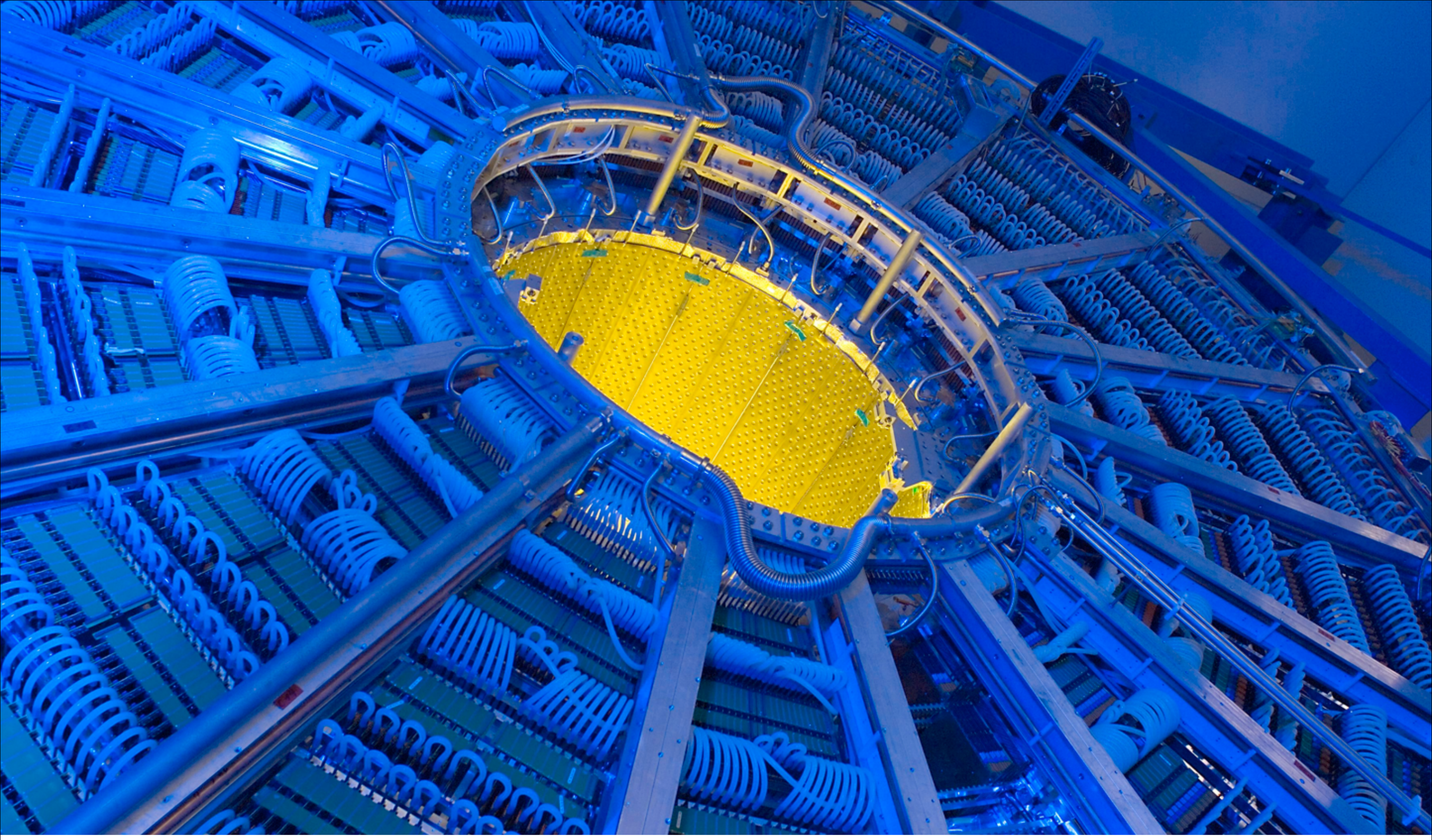


nanoAOD and Derived Data

M Floris
J F Grosse-Oetringhaus
Offline Week, March 20, 2014

Outline

- Use case:
  - The analyzer wants a **custom format** (to be analyzed with the analysis framework) containing only a very **limited subset** of the information
  - He defines the **few variables** he needs for tracks (which pass some **track cuts**) in events (which pass some) **event cuts**
  - An analysis task is run on the LEGO trains and produces the derived dataset (**nAOD**)
  - The nAOD is analyzed **locally** (if small enough) or on **CAF** or with the **trains**
  - When not needed any more, the nAOD is deleted: no centralized bookkeeping (**lifecycle** managed by PWGs)
    - Strict PWG disk quotas to be enforced

Some groups already use nanoAODs for specific cases (e.g. PWG-DQ, hyper-nuclei)
👍 More optimized / 👎 Need (at least) a custom replicator (= 2,3 additional classes)

Thanks for discussions, feedback and help:
Ramona, Stefania, Andreas, Peter, Andrei, Leonardo, Marco, Michael…

The big picture

AliAnalysisTaskFilterNanoAOD
AliAnalysisCuts *trk,*evt TString varList
AddFilteredAOD

The big picture

AliAnalysisTaskFilterNanoAOD

AliAnalysisCuts *trk,*evt

TString varList

AddFilteredAOD

AliAODHandler

AliAODExtension

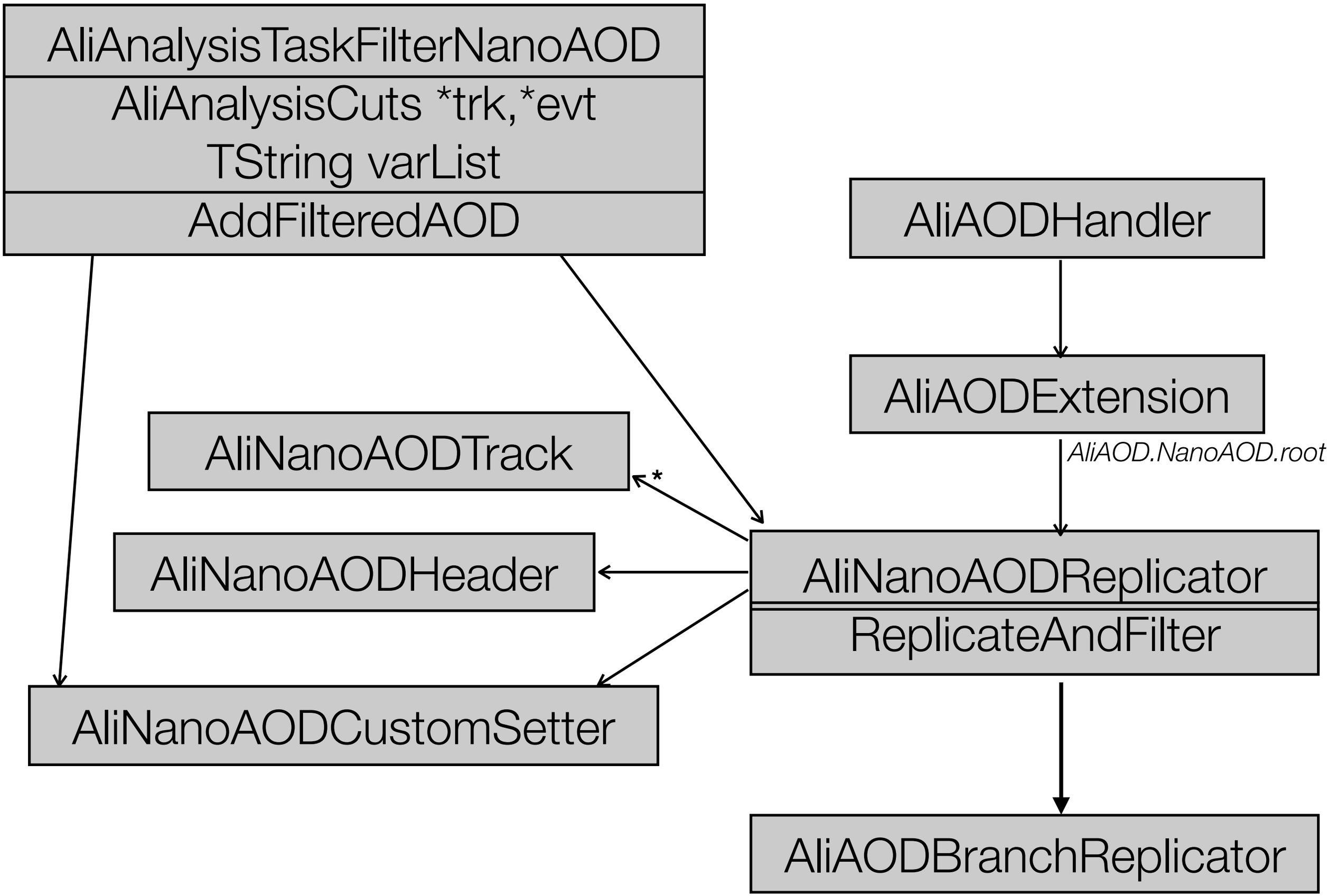
AliAOD.NanoAOD.root

AliNanoAODReplicator

ReplicateAndFilter

AliAODBranchReplicator

The big picture



The big picture

AliAnalysisTaskFilterNanoAOD

AliAnalysisCuts *trk,*evt

TString varList

AddFilteredAOD

AliAODHandler

AliAODExtension

AliAOD.NanoAOD.root

AliNanoAODTrack

*

AliNanoAODHeader

AliNanoAODReplicator

ReplicateAndFilter

AliNanoAODCustomSetter

Singleton

AliNanoAODTrackMapping

AliAODBranchReplicator

AliAODNanoTrack

Light weight track, for analysis-specific AOD
Only a subset of the info copied from AOD/ESD track

```
new AliNanoAODTrack(aodTrack, "pt,theta,phi,cstBayesProb")
```

Can be an AOD or ESD track

Only those variables are copied to the special track

Custom Variables can be defined to store derived quantities (e.g. PID probabilities)

Interface similar to AliAODTrack, derives from AliVTrack (abundant use of AliFatals)

Implementation of tracks and headers

```
Int_t    fNVars;     // Number of kimematic variables, set by constructor
std::vector<Double32_t> fVars; // Array of kinematic vars
```

Variables are stored internally into 1 array.
The array is written as a single branch in the aodTree
(We may have understood how to split it, to be investigated)

```
virtual Double_t Pt() const { return GetVar(AliAODSpecialTrackMapping::GetInstance()->GetPt()); }
```

The getters find out where each variable is stored using a static mapping class which is written to the UserInfo of the aodTree.

```
Double_t GetVar(Int_t index)  const {
  if(index>=0 && index < fNVars) return fVars[index];
  AliFatal(Form("Variable %d not included in this special aod", index));
  return 0;}
```

The basic getter checks if the requested variable was actually included in the derived AOD, if not it AliFatals

Implementation of tracks and headers

```
Int_t    fNVars;     // Number of kimematic variables, set by constructor
std::vector<Double32_t> fVars; // Array of kinematic vars
```

Variables are stored internally into 1 array.
The array is written as a single branch in the aodTree
(We may have understood how to split it, to be investigated)

```
virtual Double_t Pt() const { return GetVar(AliAODSpecialTrackMapping::GetInstance()->GetPt()); }
```

The getters find out where each variable is stored using a static mapping class which is written to the UserInfo of the aodTree.

```
Double_t GetVar(Int_t index)  const {
  if(index>=0 && index < fNVars) return fVars[index];
  AliFatal(Form("Variable %d not included in this special aod", index));
  return 0;}
```

The basic getter checks if the requested variable was actually included in the derived AOD, if not it AliFatals

Implementation of tracks and headers

```
Int_t    fNVars;     // Number of kimematic variables, set by constructor
std::vector<Double32_t> fVars; // Array of kinematic vars
```

Variables are stored internally into 1 array.
The array is written as a single branch in the aodTree
(We may have understood how to split it, to be investigated)

```
virtual Double_t Pt() const { return GetVar(AliAODSpecialTrackMapping::GetInstance()->GetPt()); }
```

The getters find out where each variable is stored using a static mapping class which is written to the UserInfo of the aodTree.

```
Double_t GetVar(Int_t index)  const {
  if(index>=0 && index < fNVars) return fVars[index];
  AliFatal(Form("Variable %d not included in this special aod", index));
  return 0;}
```

The basic getter checks if the requested variable was actually included in the derived AOD, if not it AliFatals

Alternatives?

All other alternatives imply writing custom nano classes for each analysis.
Can be done in a smart way: code compiled on the fly, track definition stored in output root file (suggestion from Ruben)
Better optimization, more code. To be thought of.

In Practice

Filter configuration

```
__R_ADDTASK__->SetVarList("pt,theta,phi,cstNSigmaTPCPi,cstNSigmaTPCKa,cstNSigmaTPCPr");
__R_ADDTASK__->SetVarListHead("cstCentr,cstQVec");
AliESETrkCut * trk = new AliESETrkCut; // Derives from AliAnalysisCuts
AliESEEvtCut * evt = new AliESEEvtCut; // Derives from AliAnalysisCuts
AliAnalysisESESetter* setter  = new AliAnalysisESESetter;
__R_ADDTASK__->SetTrkCuts(trk);
__R_ADDTASK__->SetEvtCuts(evt);
__R_ADDTASK__->SetSetter(setter);
```

- Track and event cuts can be custom or use basic class which will be provided
  - From ESDs, can use AliESDtrackCuts
- Cuts can be streamed to separate output file if needed (e.g. cuts statistics).
- Setter can be null if there are no cst variables

Task changes

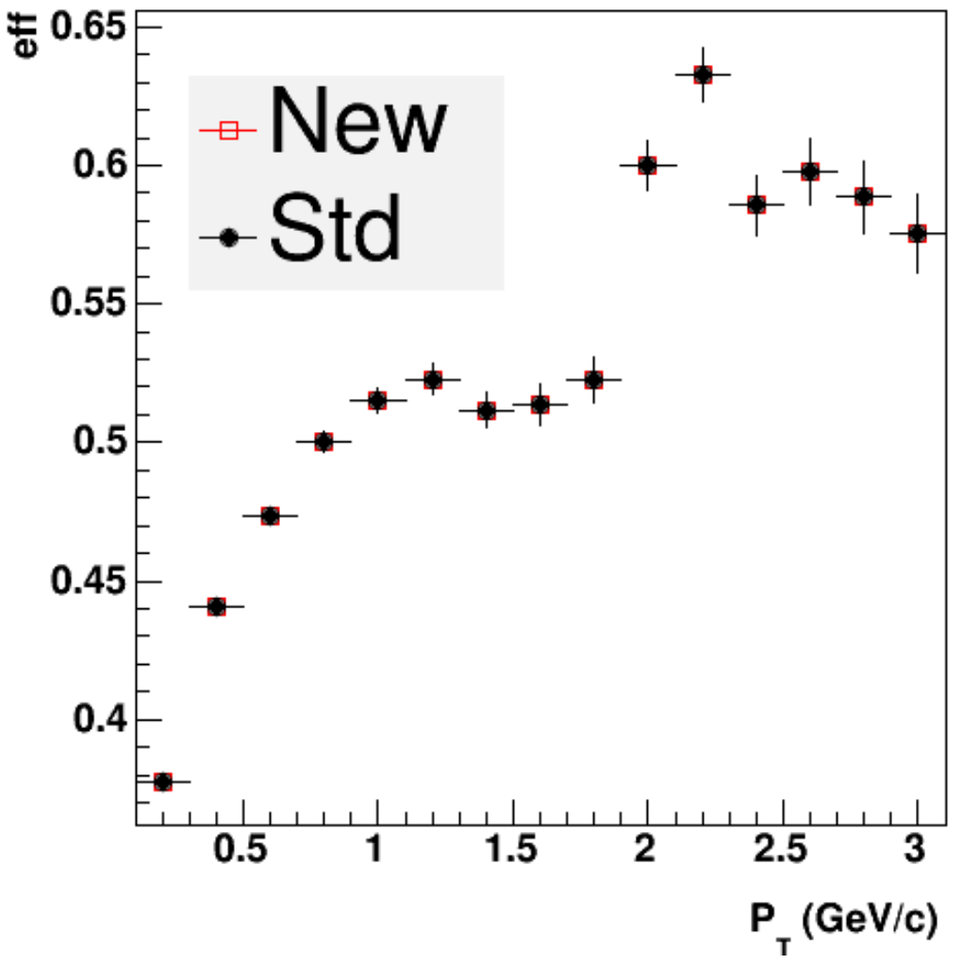
```
#ifdef GOOD_OLD_AOD
      AliAODTrack* track = (AliAODTrack*) event->GetTrack(i);
#else
      AliAODSpecialTrack* track = (AliAODSpecialTrack*) event->GetTrack(i)
#endif
```

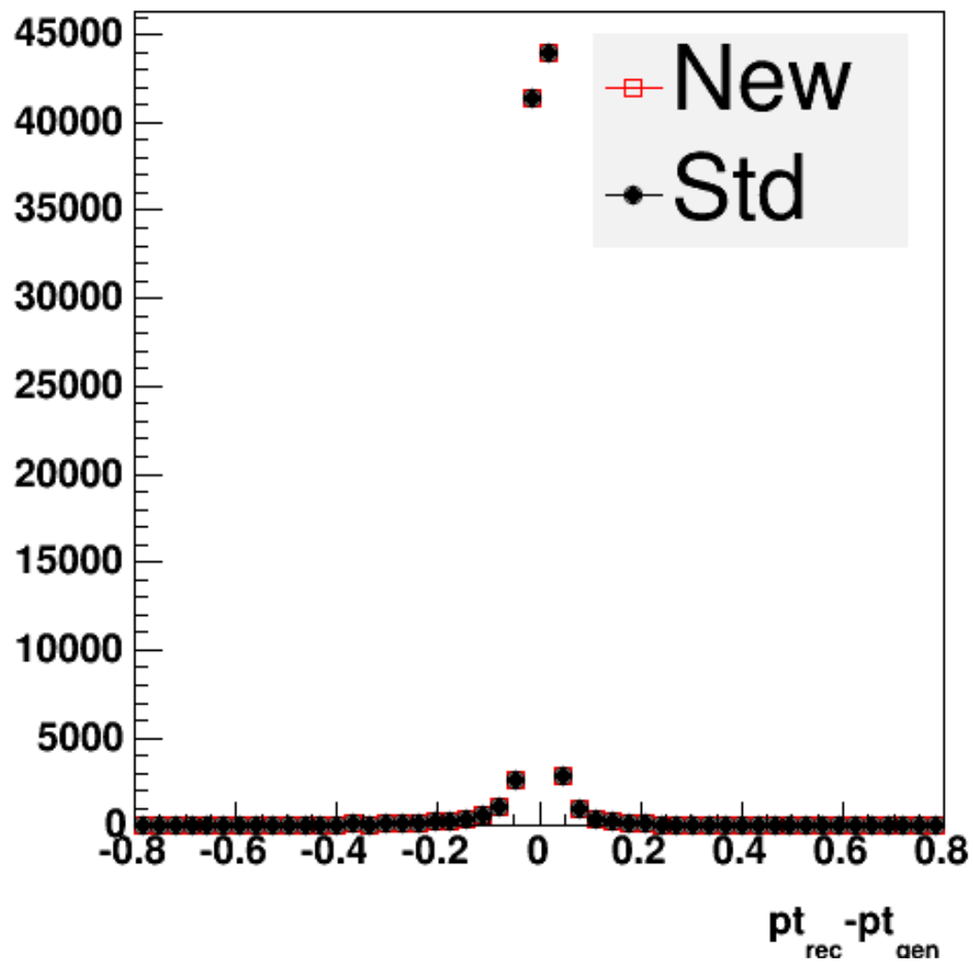
- will not always be this trivial: derived custom vars
- #ifdef may not be needed using the virtual interfaces, see later

Performance and tests

Test	Standard	Special
Filter only data, no trk/evt cuts	5.6 MB	417 KB
Data and MC, no trk/evt cuts	17 MB	2.1 MB

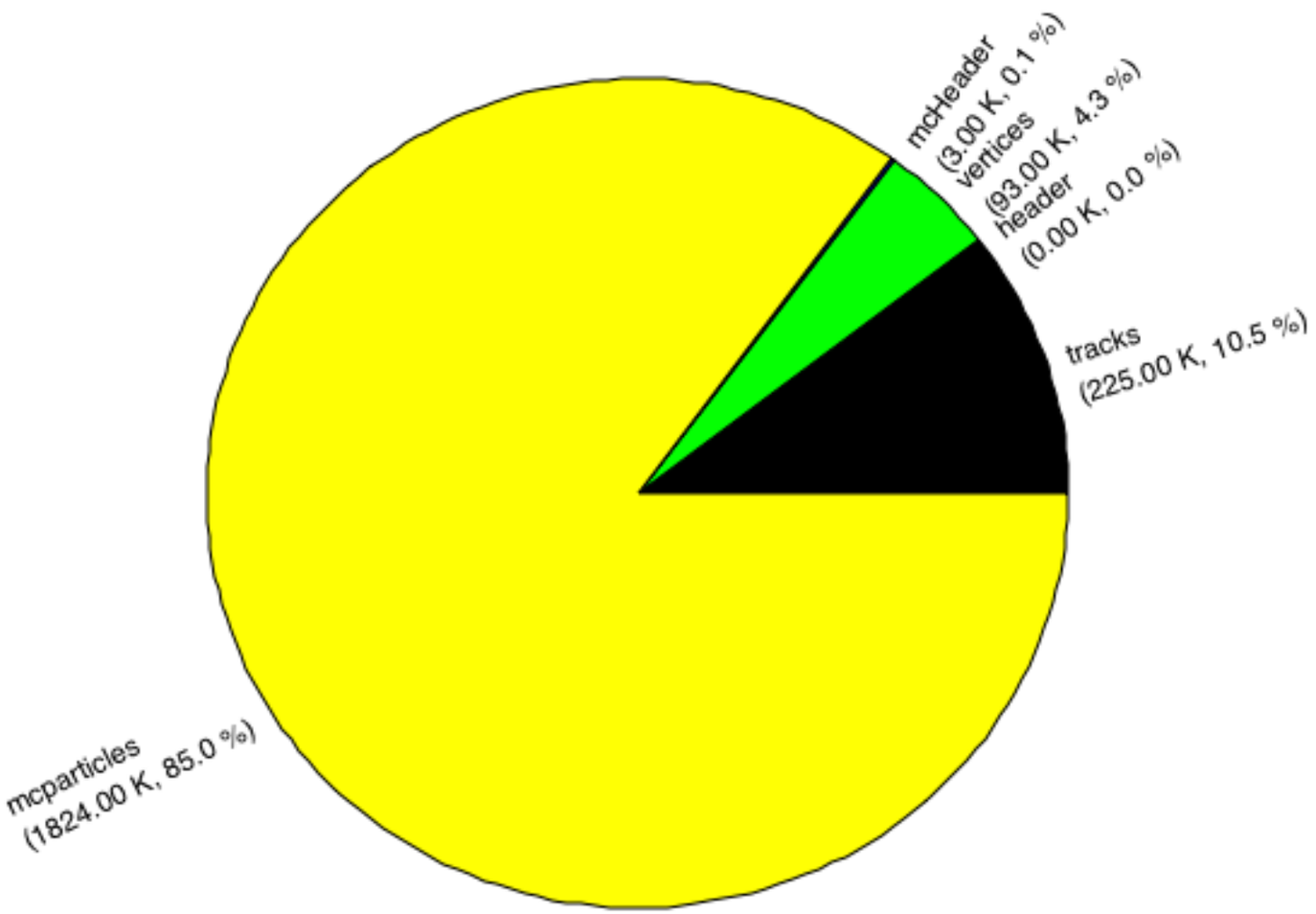
MC: same AliMCPartilces, keeps only charged primaries + all particles associated to a track and their ancestors



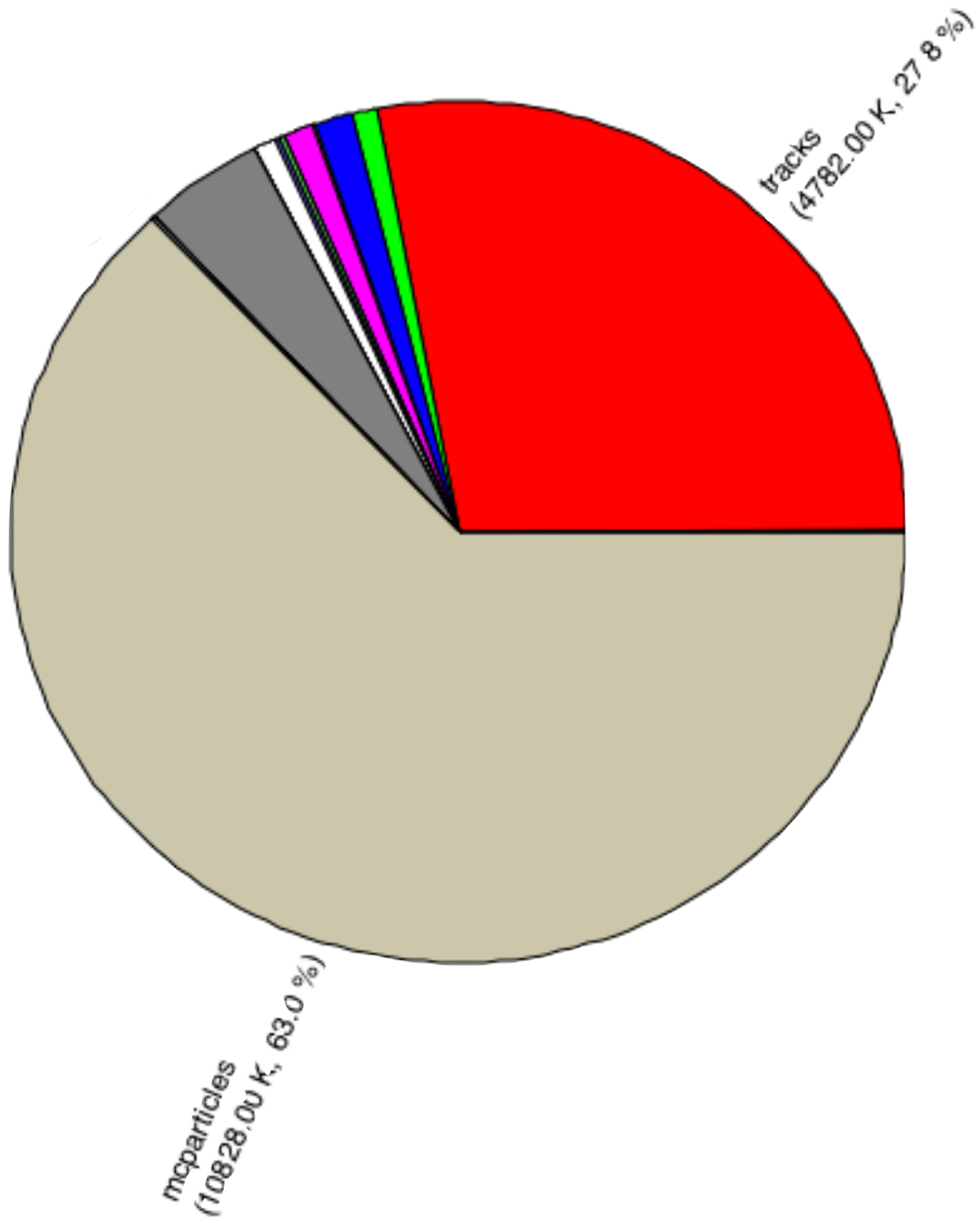


One chunk

nano AOD



standard AOD



Tracks: 4782 K →225 K (32 B per nanoAOD track, 10 vars, CF = 1.4)
MC Particles: 10828 K → 1824 K (22 B per MC particle)
(Only charged primaries + particles originating tracks and their ancestors)

Filtering performance (Real use cases)

Tests on the LEGO Trains

	Test	Standard	Nano	Reduction	Wall Time
LHC10h subset	**PWG-LF Spectra ESE Data**	577.7 GB	18.82 GB	31.7	1d 23:10
	PWG-CF Correlations Data	551.5 GB	17.95 GB	30.7	1d 15:17
LHC11a10a_bis	**PWG-LF Spectra ESE MC**	1.291 TB	138.8 GB	9.3	10d 20:18

PWG-LF-ESE MC, Analysis of nano on trains

- **Nano**
  - Totals: running time: **23:27** | output size: 967.3 MB
  - Files/job (for done jobs): min: 1, max: 1, average: 1 , standard deviation: 0
    - Could be run with more than 1 file/job. To be tested.
  - Running time/job (for done jobs): min: 0m 3s, max: 29m, **average: 3m 2s**, standard deviation: 4m 23s, 95% done after 11m 36s
- **Std**
  - Totals: running time: **6d 6:57** | output size: 1.143 GB
  - Files/job (for done jobs): min: 1, max: 100, average: 38.2 , standard deviation: 40.3
  - Running time/job (for done jobs): min: 0m 6s, max: 5:50, **average: 19m 23s**, standard deviation: 42m 34s, 95% done after 1:03

PWG-LF-ESE MC, Analysis of nano on trains

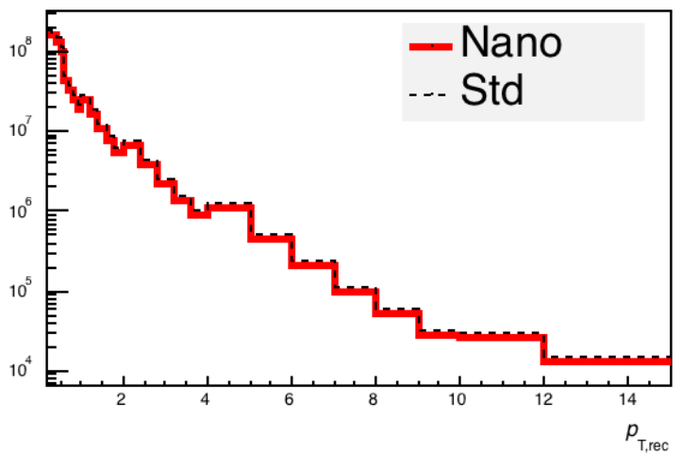
Nano

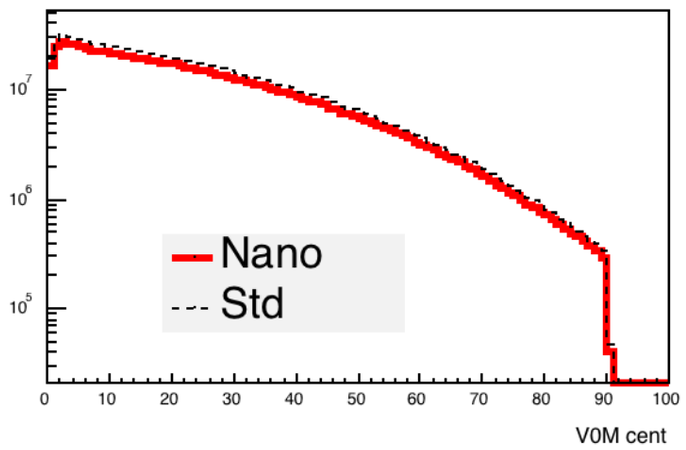
Job Overview						
State	Jobs	Files	Input size	Files/job		
				min	max	avg
DONE	464	464	136.8 GB	1	1	1
ERROR_V	3	3	2.043 GB	1	1	1
ERROR_E (TTL)	0	0	0 B	0	0	0
ERROR_E (mem)	0	0	0 B	0	0	0
ERROR_E (disk)	0	0	0 B	0	0	0
ERROR_EW	0	0	0 B	0	0	0
Other	0	0	0 B	0	0	0

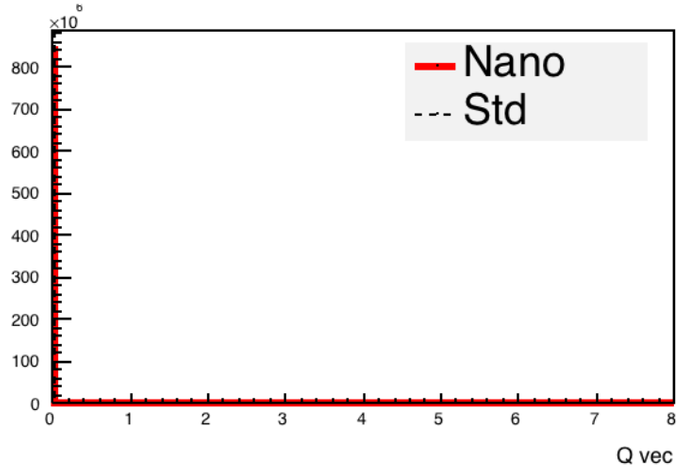
Std

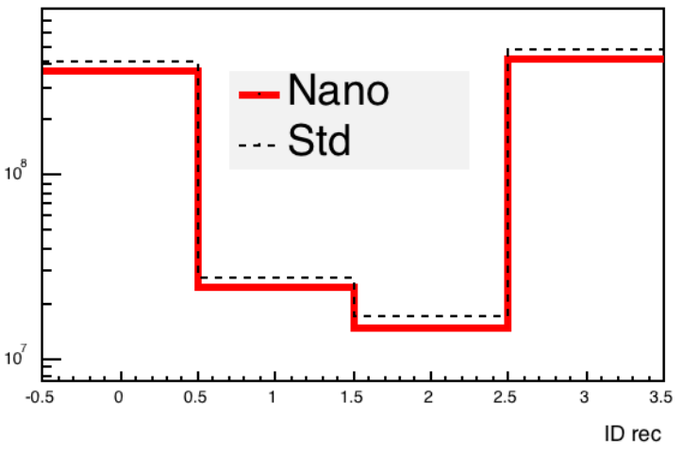
Job Overview						
State	Jobs	Files	Input size	Files/job		
				min	max	avg
DONE	467	17829	1.326 TB	1	100	38.2
ERROR_V	1	100	8.826 GB	100	100	100
ERROR_E (TTL)	0	0	0 B	0	0	0
ERROR_E (mem)	2	200	21.61 GB	100	100	100
ERROR_E (disk)	0	0	0 B	0	0	0
ERROR_EW	9	423	21.31 GB	2	100	47
Other	0	0	0 B	0	0	0

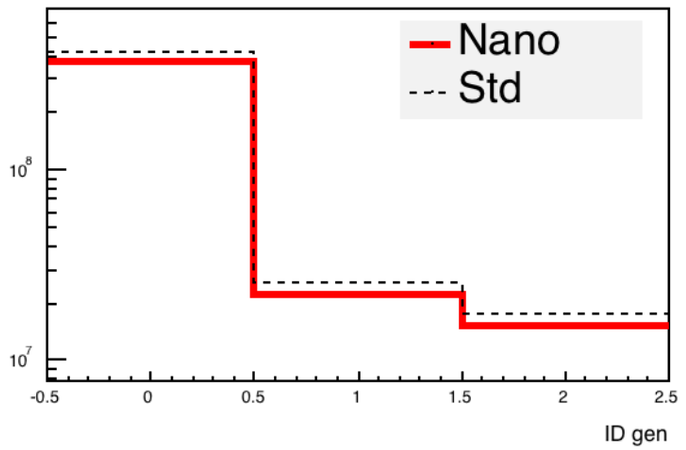
PWG-LF-ESE MC, Results

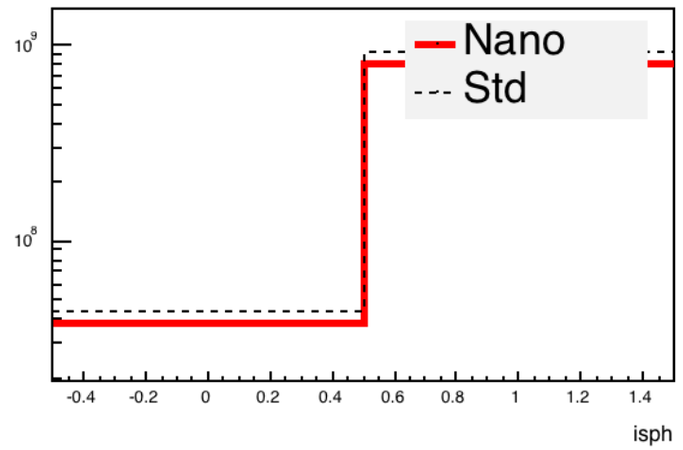


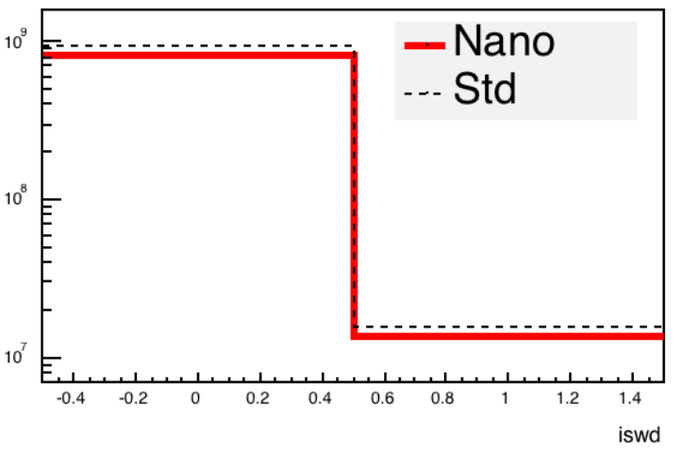


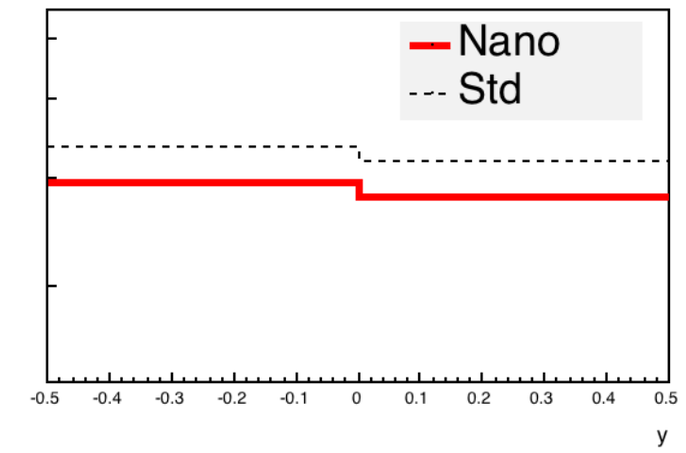






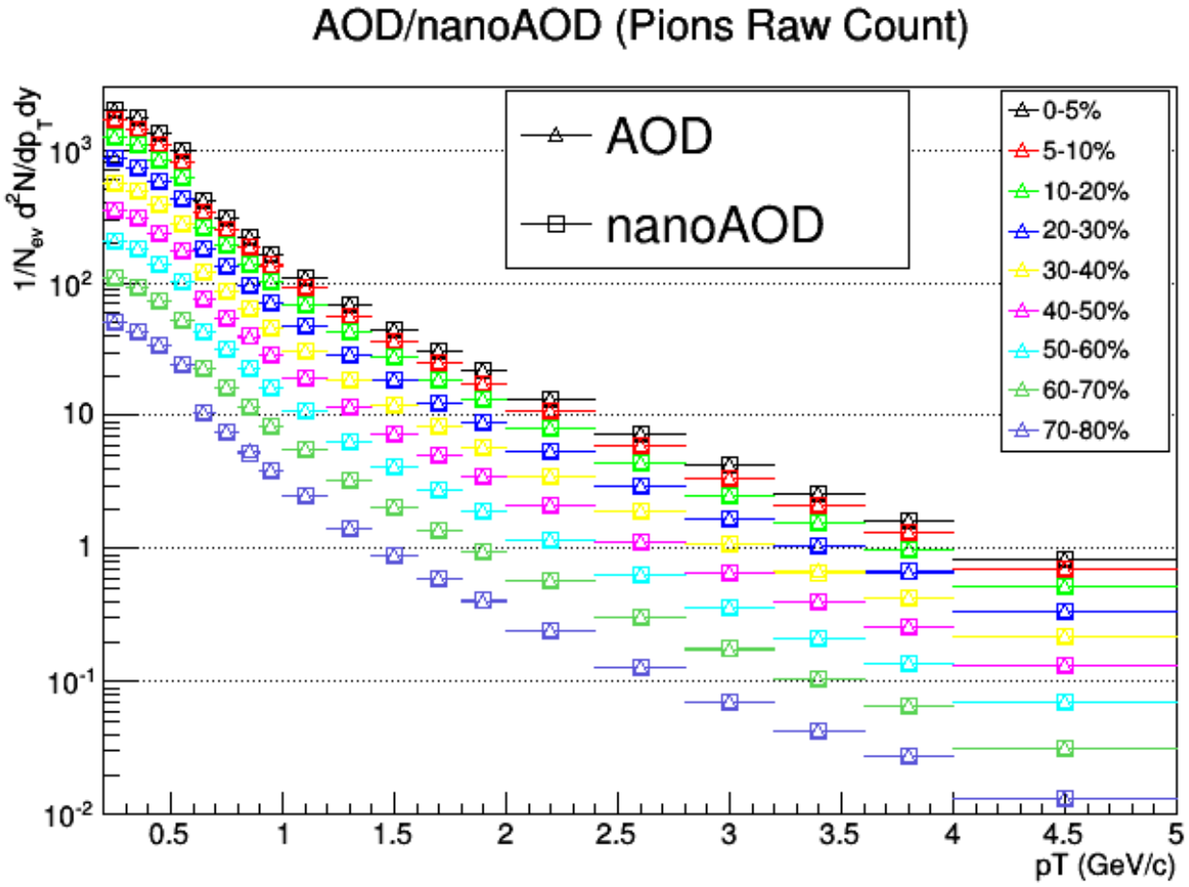


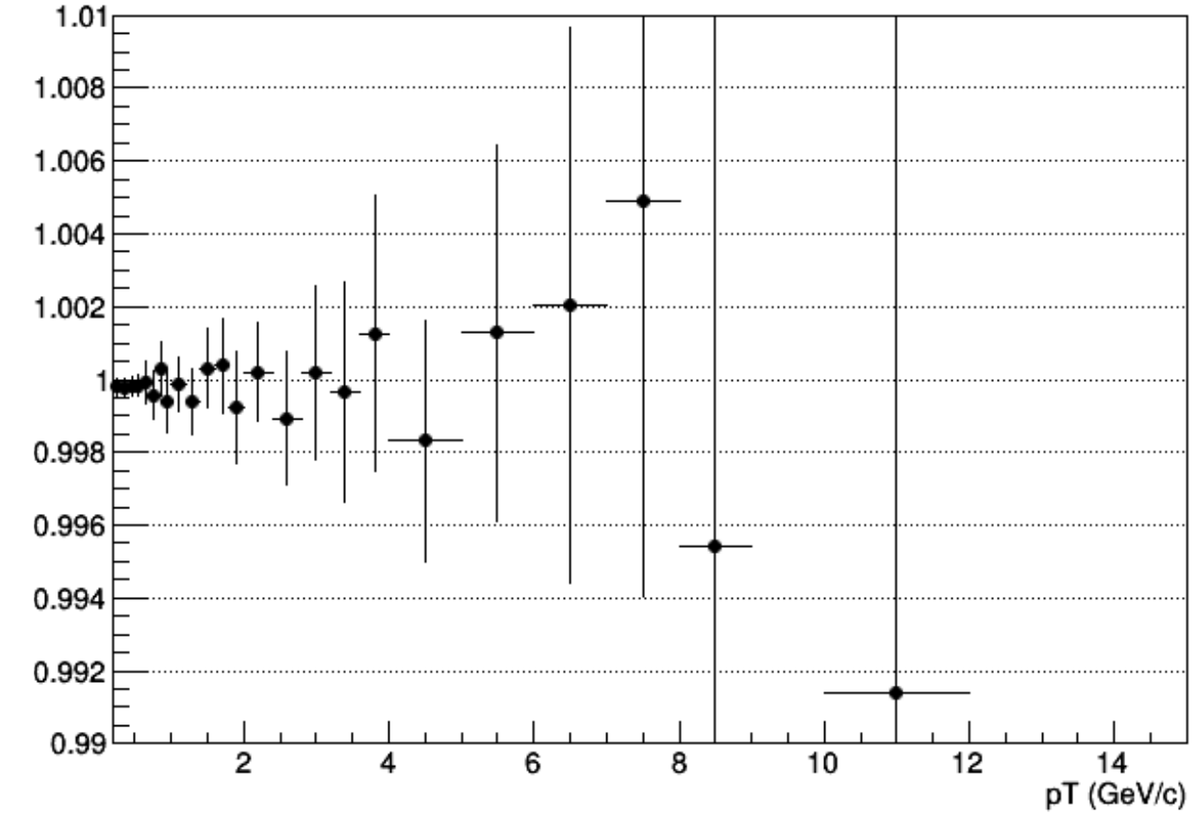


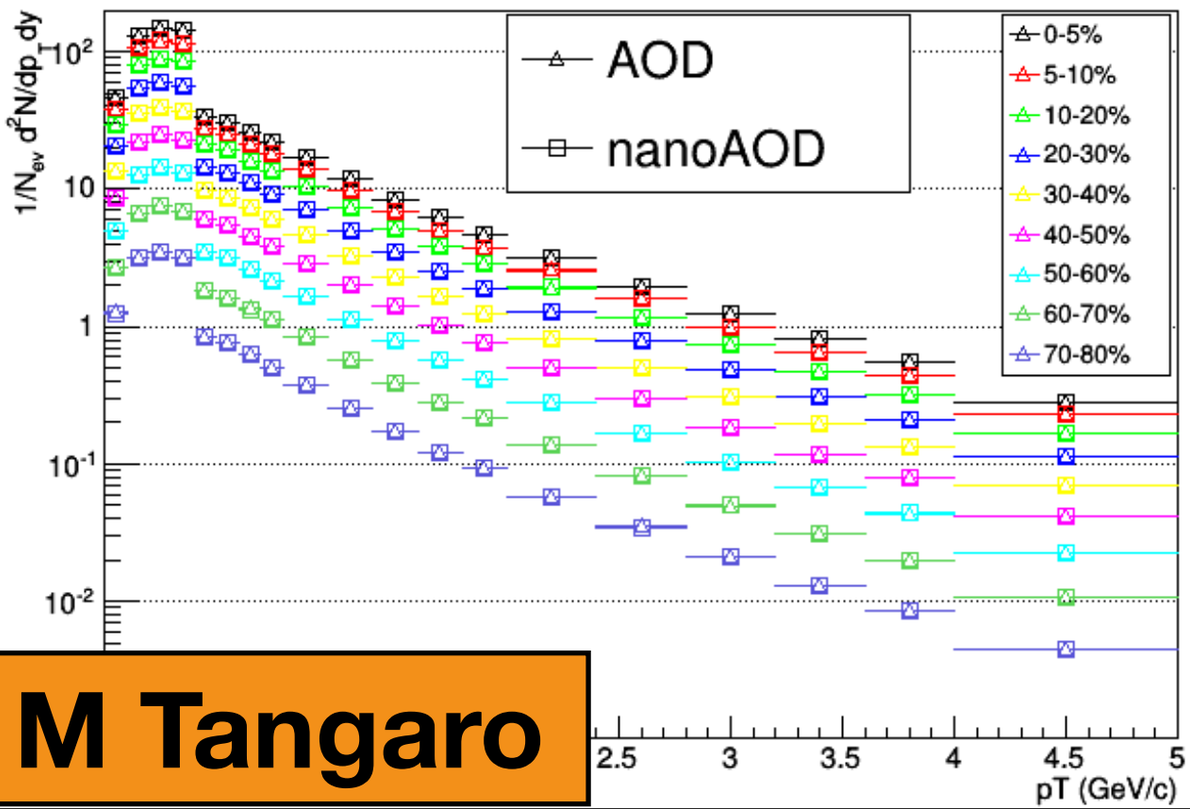


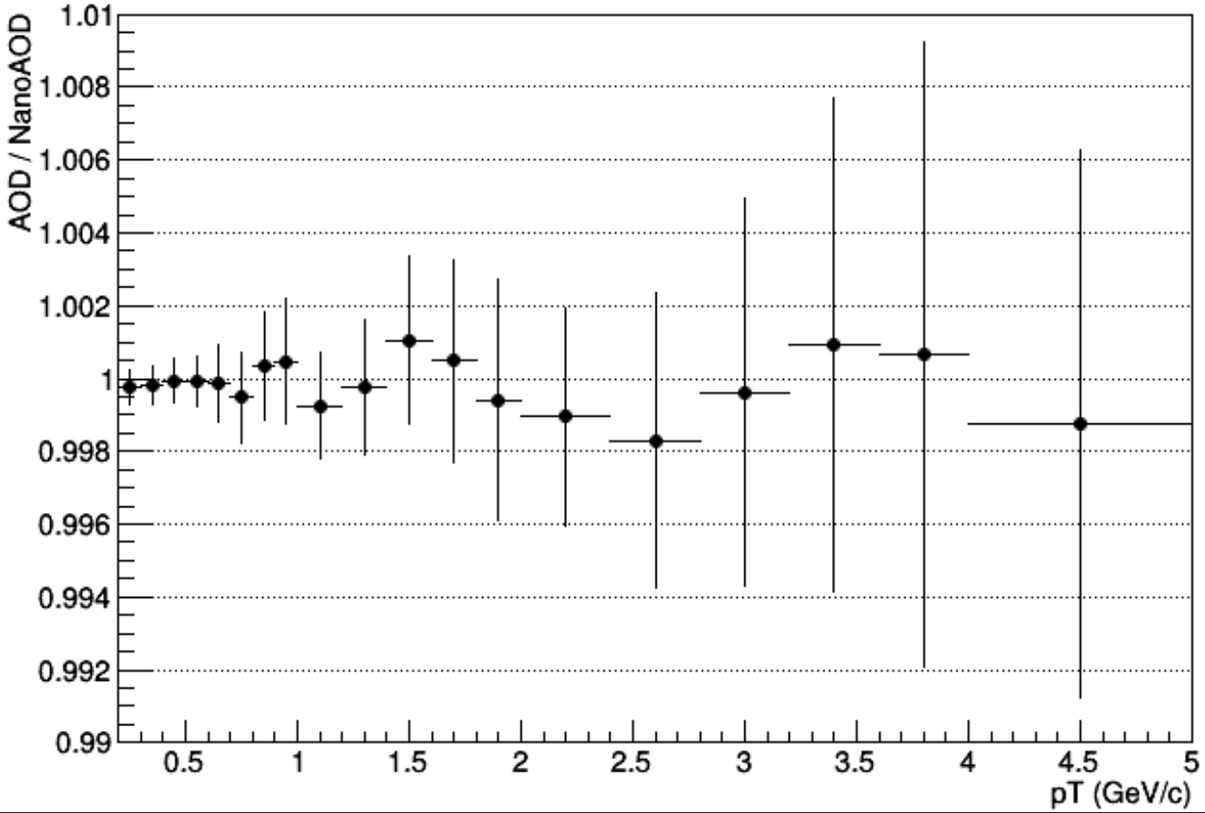
Slightly larger statistics processed in std, compatible results

PWG-LF-ESE MC, Results



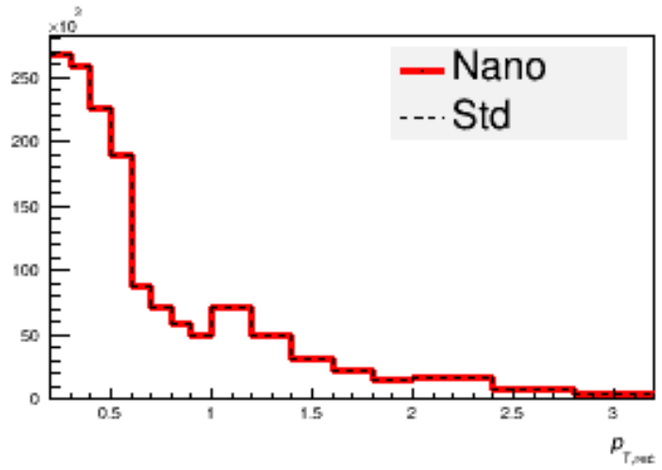


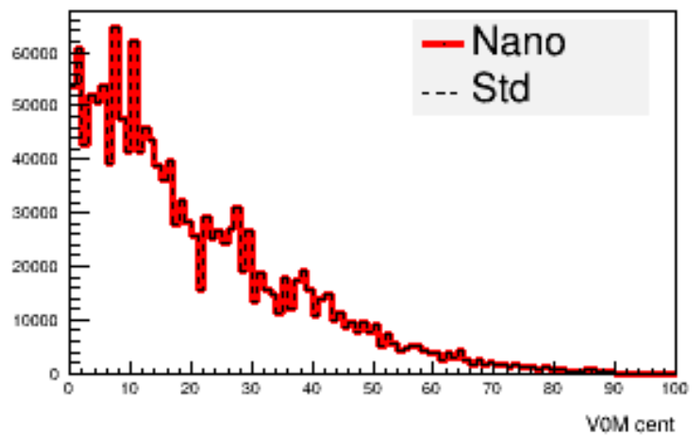


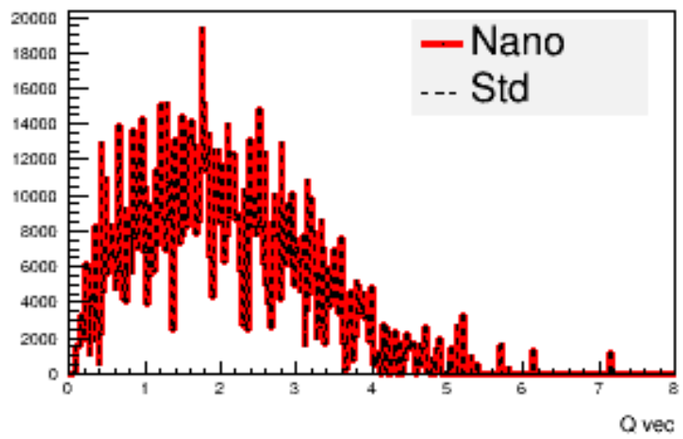


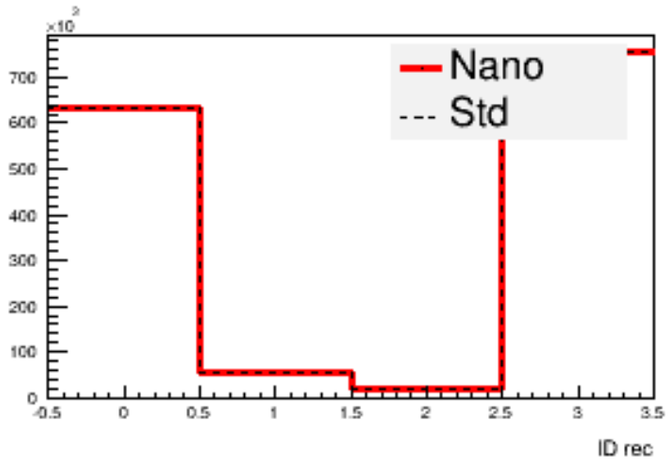
M Tangaro

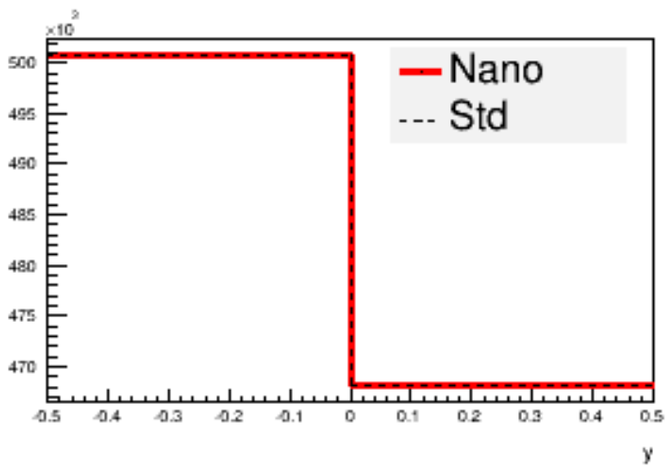
PWG-LF-ESE LHC10h, 3 files (2700 evts)

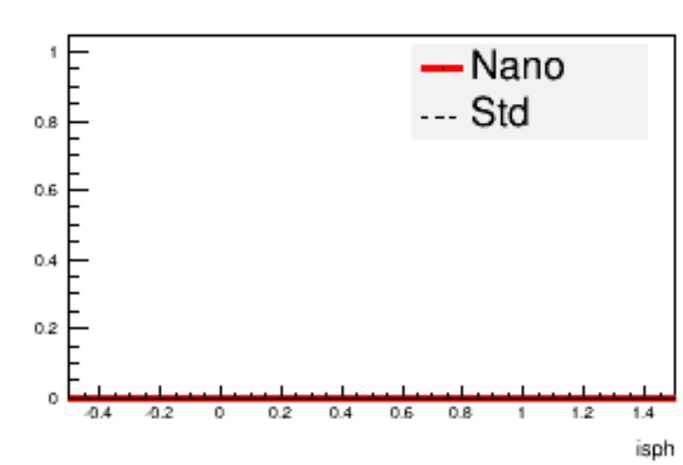












Standard:

```
1326 M
init time:          1.29961[sec]
I/O & data mng.:70.4448 [sec]
task execution: 11.5753 [sec]
total time:    CPU=80.68 [sec]
               REAL=83.3197[sec]
```

Nano:

```
50 M (1/26 * std)
init time:          0.367178[sec]
I/O & data mng.: 2.63543 [sec]
task execution: 0.673858 [sec]
total time:       CPU=3.68 [sec]
                  REAL=3.67646[sec]
                  (1/22*std)
```

⇒The whole LHC10h: ~340 GB, Processed in 7 h*core
(this analysis run for a total of 172 days over the last month)

PWG-CF, analysis on Lego trains

CF nanos (extrapolated to LHC10h AOD086): **9.5 TB** (AOD) → **310 GB** (nanoAOD)

CPU efficiency: **97.3%** (AOD) → **98.5%** (nanoAOD)

Single job on Δϕ correlations (very **CPU intensive**):

AOD: Real time 0:32:10, CP time 1877.860

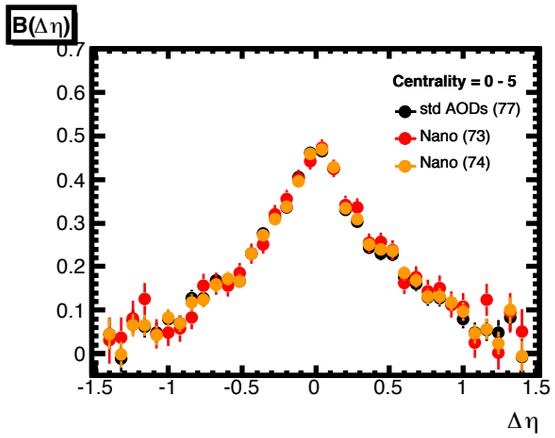
NanoAOD: Real time 0:20:22, CP time 1217.910

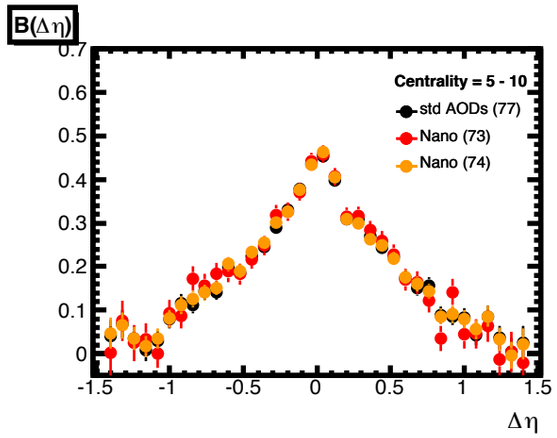
⇒ **33% less**

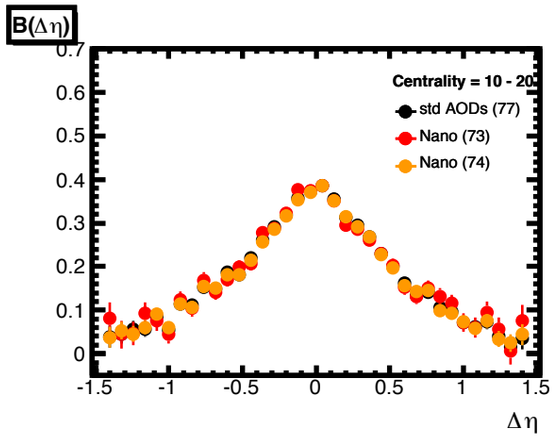
Scaled to LHC10h AOD086 (90 runs), running time: 1y 185d reduces to 1y (these trains run ~10 times per month only in CF)

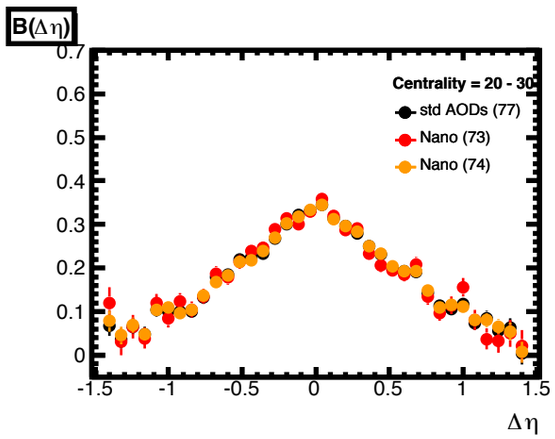
PWG-CF, Balance function, data

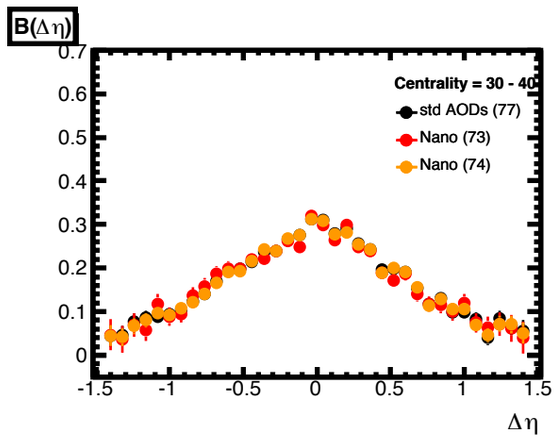
Delta Eta

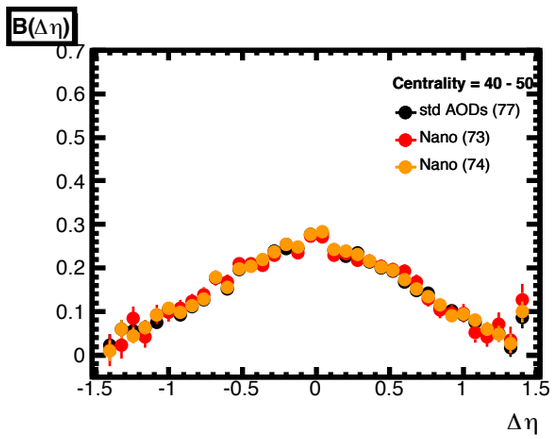


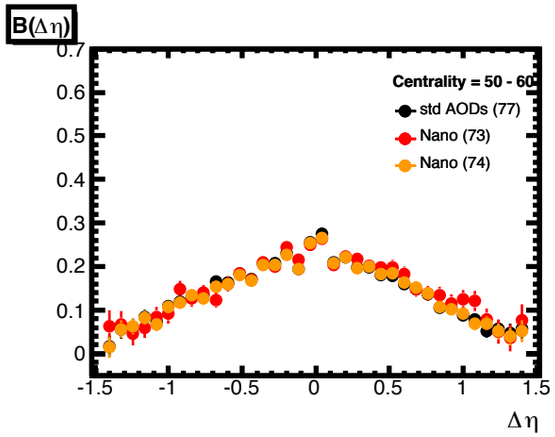


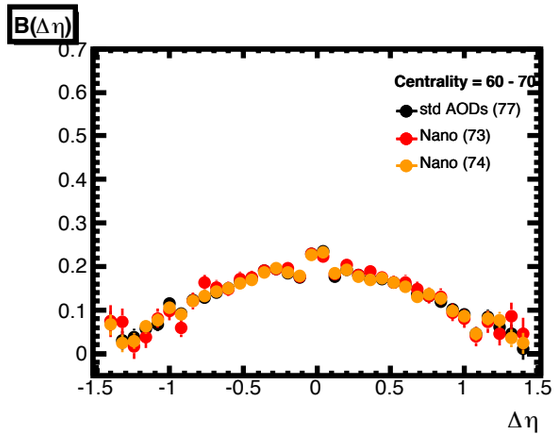


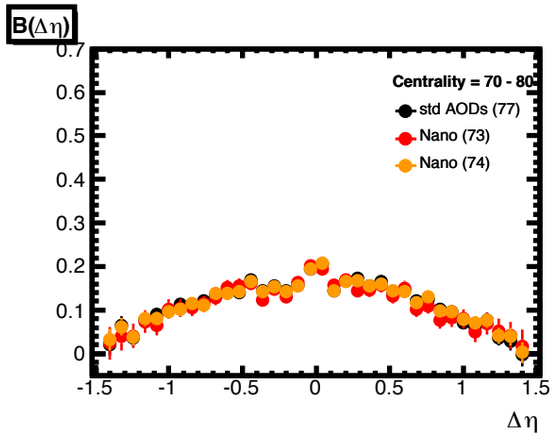












M Weber

Summary

- Fully **configurable nanoAOD** prototype developed
- Size: **10-30 times** smaller than a standard AOD (can try to further improve MC)
- For **I/O bound** analyses (spectra), similar gain in wall time (6-20 times faster)
  - Huge gain on the trains
  - Even more, Some analyses could be run locally on a powerful-enough desktop [the Spectra ESE analysis would require ~500 GB (data+MC)]
- For **CPU bound** analyses (correlations) gain is less obvious
  - ~33% faster
  - Can nevertheless help to reduce confusion (e.g. produce a hybrid tracks sample)
- It may not be the best solution for all analyses, but it can surely significantly improve some
  - We suggest to put it into production!

TODO 1: Analysis framework

- Integration with the analysis framework to be improved
- At the moment, tests done using the same analysis AliAODEvent, but **recasting** AliAODTrack* to AliNanoAODTrack* (similar for header)

```
AliAODTrack  *GetTrack(Int_t nTrack)
```

- **Proposal**: change AliAOD{Track,Header}* to AliV{Track,Header}* in **AliAODEvent**
  - Would require patching some analysis tasks and framework classes
  - If a user can rely on the V track interface, no changes are needed in the task between processing AOD or nanoAOD format
  - If one needs specific methods, the tracks will have to be explicitly recasted.
- **Alternatives**
  - Write custom AliNanoAODEvent
    - Lots of duplicated code, the NanoAOD can still contain standard branches (e.g. mc particles)
    - Could be addressed deriving AliNanoAODEvent from AliAODEvent. To be investigated
  - Add a "GetNanoAODTrack" interface to "AliAODEvent"
    - More invasive changes to user tasks

TODO 2: PID Framework + Code clean-up

- At the moment the (e.g. in the Spectra ESE case) **PID quantities** are stored as custom variables

- Possible alternatives

  - Standardize the most common ones (e.g. nsigmas) and adapt **PID framework** so that it recognizes Nano AODs

  - Standardize the most common ones and use another **interface class** (e.g. PWG/Tools/AliHelperPID.h)

- **NB**: we have a working prototype, but a lot of clean-up of the code + interfacing work is needed!

  - Before continuing, we have to be (reasonably) sure that it will be used

  - Idea presented to PB: encouraged to finish the development and test it in a real production environment (provided no technical showstopper is identified by offline)

Some QA (Train 77, std AOD)

- **Std AOD (77):**
  - **Train run finished** at: today 07:55 (train duration: 21:25)
  - **Totals:** running time: 38d 21:54 | output size: 16.85 GB
  - **Files/job** (for done jobs): min: 1, max: 7, average: 6.3 , standard deviation: 1.7
  - **Running time/job** (for done jobs): min: 3m 8s, max: 12:28, average: 6:13, standard deviation: 2:57, 95% done after 10:58

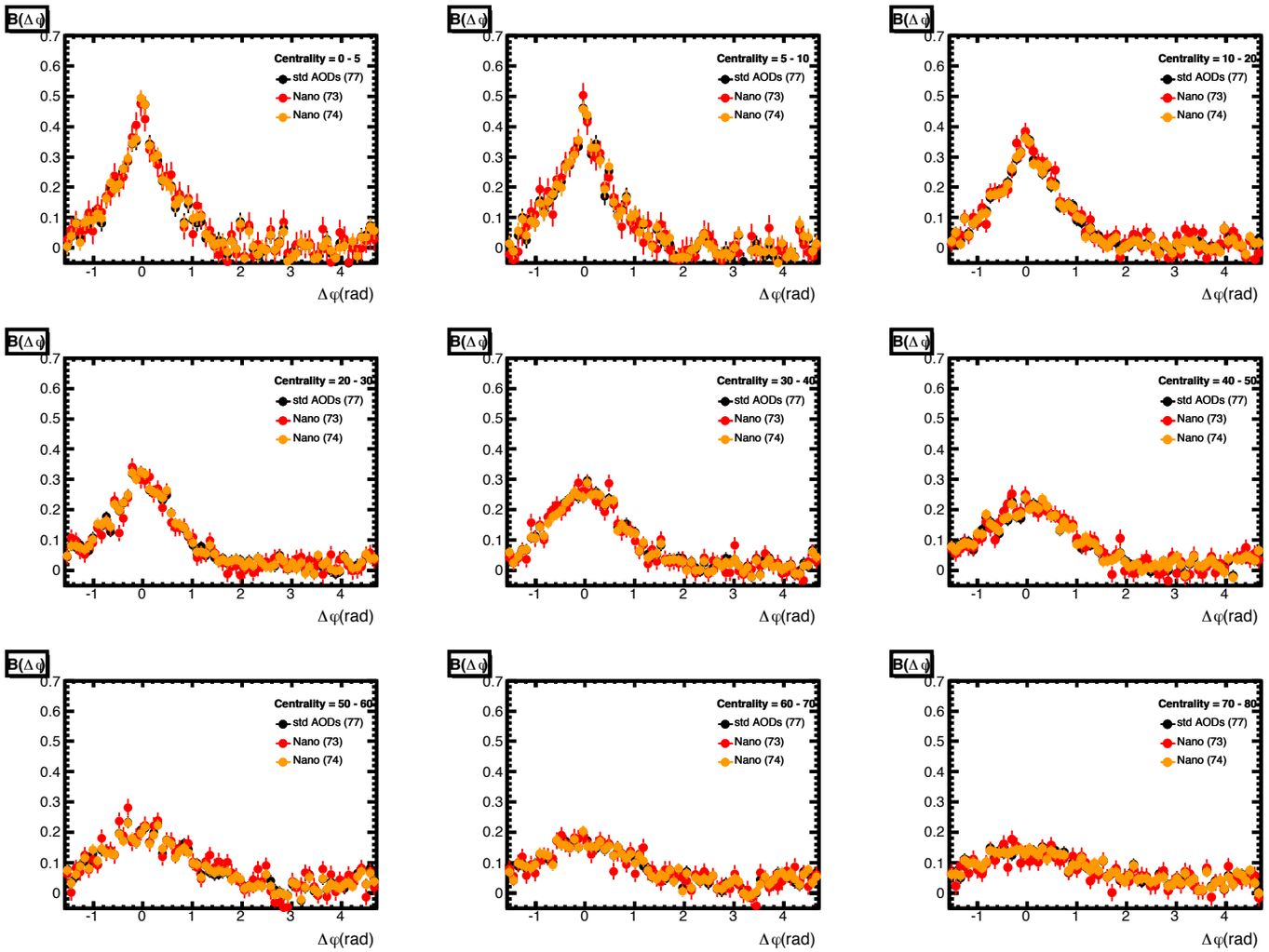
- **Nano (73) → Something was wrong with this train:**
  - **Train run finished** at: 17 Mar 2014 17:56 (train duration: 1d 6:01)
  - **Totals:** running time: 10d 15:43 | output size: 1.479 GB
  - **Files/job** (for done jobs): min: 2, max: 4, average: 3.9 , standard deviation: 0.5
  - **Running time/job** (for done jobs): min: 27m 14s, max: 12:48, average: 6:43, standard deviation: 3:11, 95% done after 12:02

- **Nano (74):**
  - **Train run finished** at: yesterday 19:55 (train duration: 10:25)
  - **Totals:** running time: 18d 15:02 | output size: 8.243 GB
  - **Files/job** (for done jobs): min: 1, max: 1, average: 1 , standard deviation: 0
  - **Running time/job** (for done jobs): min: 0m 14s, max: 5:27, average: 2:46, standard deviation: 1:34, 95% done after 5:00

WARNING: This numbers are biased (there was another task in the train)

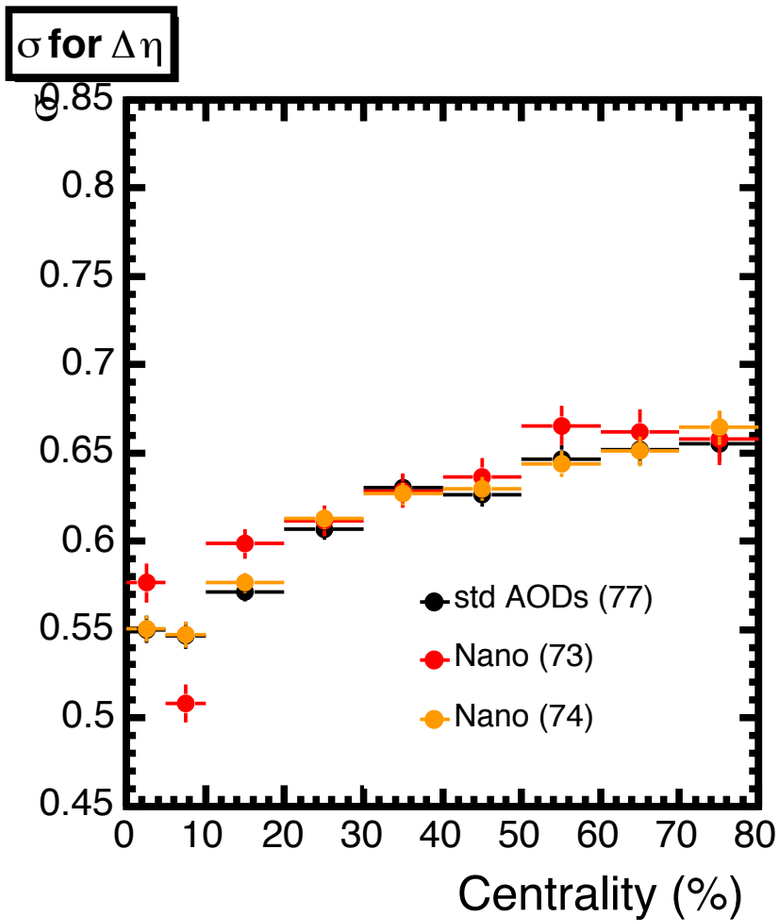
M Weber

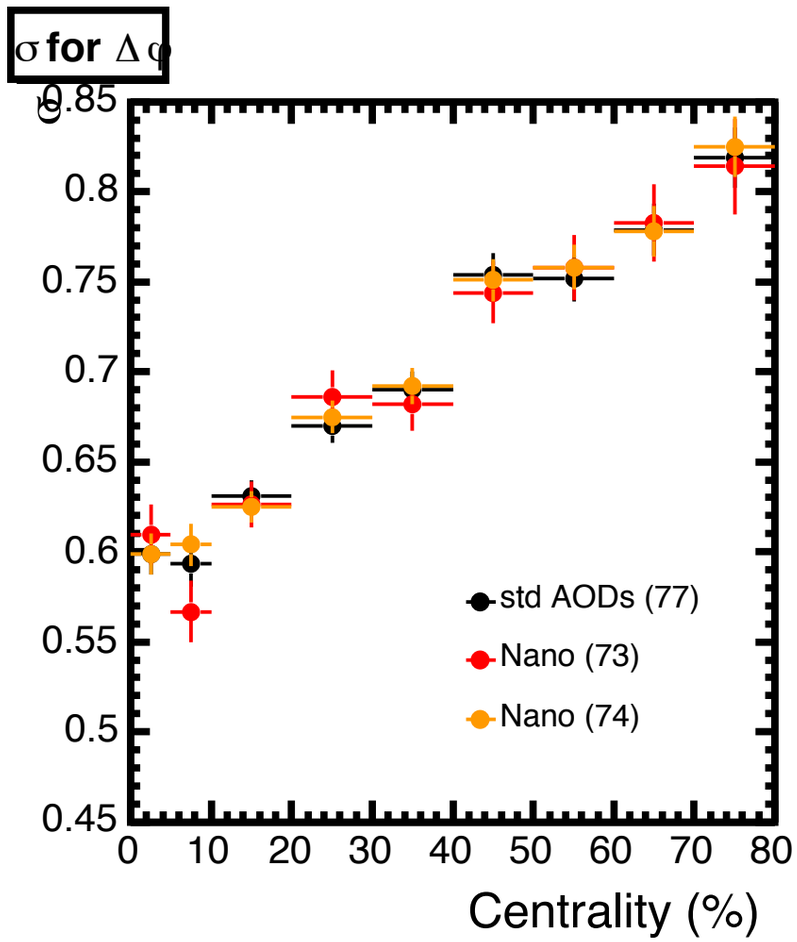
Delta Phi



M Weber

Balance function width





M Weber

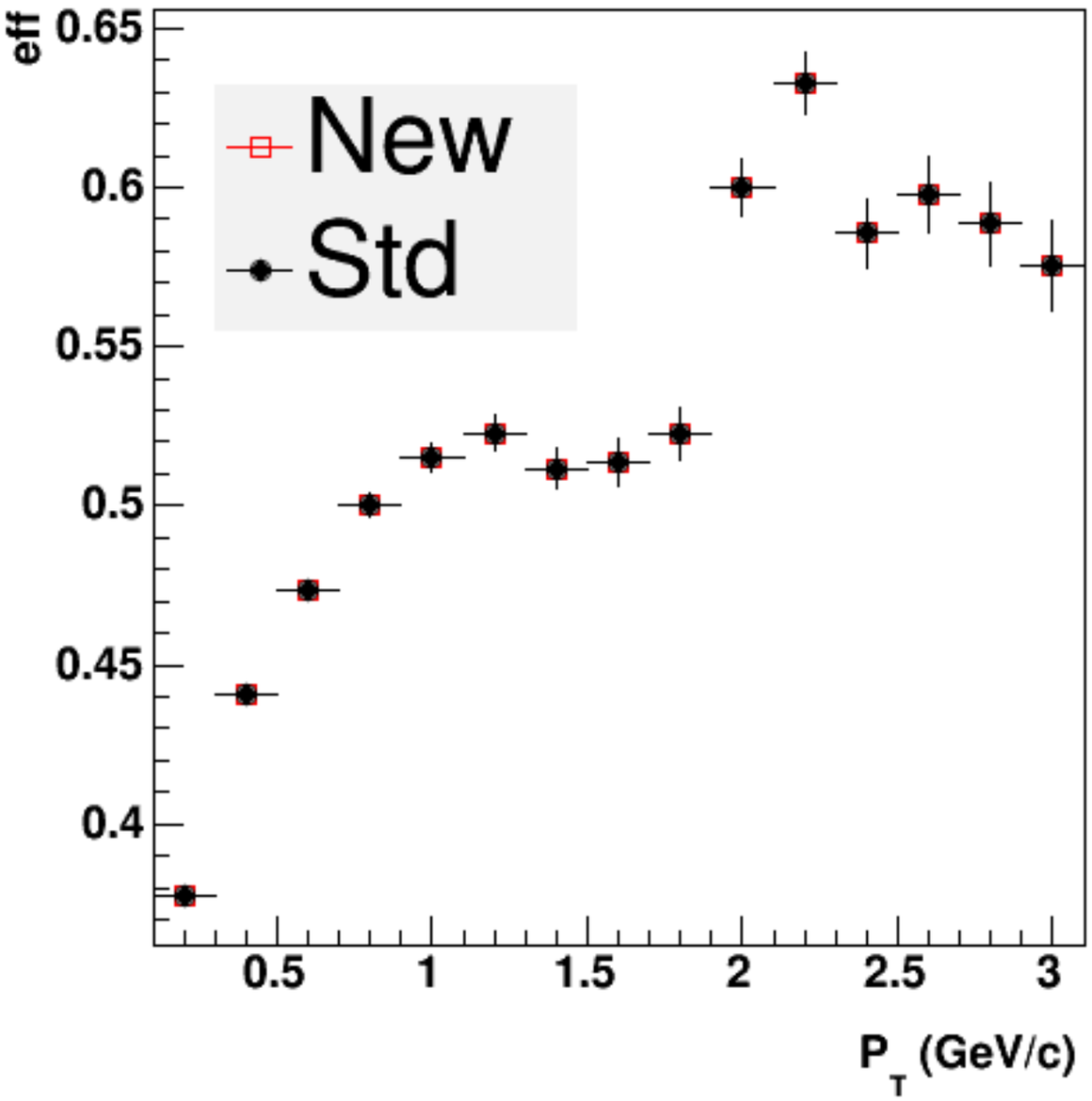
Test on LEGO Trains

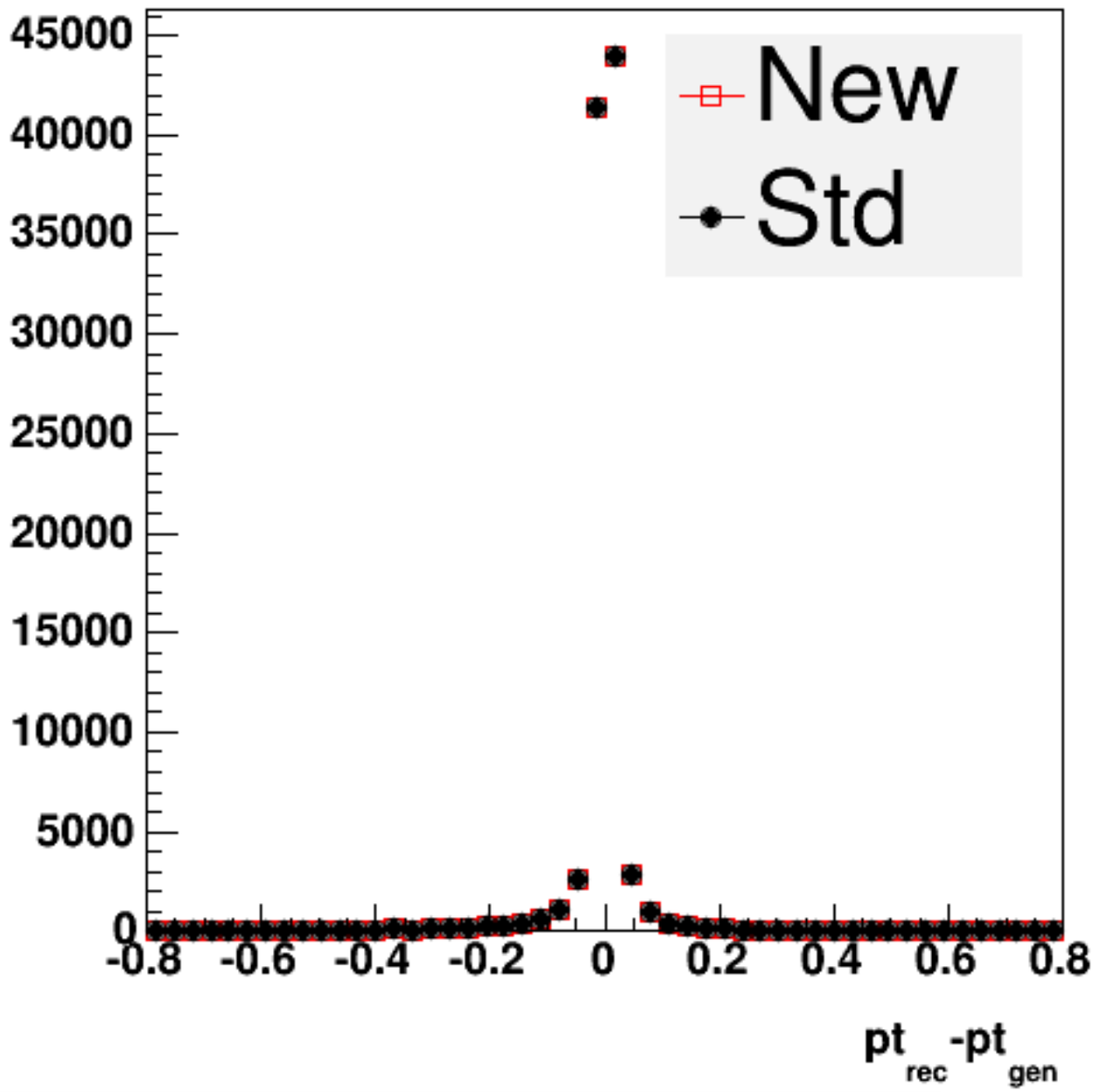
LHC11a10a_bis (PbPb 2010, MB), 93 runs, ~ 3M events

Job Overview						
State	Jobs	Files	Input size	Files/job		
				min	max	avg
DONE	464	17178	1.24 TB	1	100	37
ERROR_V	3	104	8.945 GB	1	100	34.7
ERROR_E (TTL)	0	0	0 B	0	0	0
ERROR_E (mem)	7	700	72.02 GB	100	100	100
ERROR_E (disk)	0	0	0 B	0	0	0
ERROR_EW	9	900	74.83 GB	100	100	100
Other	0	0	0 B	0	0	0

Train run finished at: today 01:55 (train duration: 3:58)
Totals: running time: 5d 12:23 | output size: 135.6 GB
Files/job (for done jobs): min: 1, max: 100, average: 37 , standard deviation: 39.5
Running time/job (for done jobs): min: 0m 6s, max: 1:55, average: 17m 7s, standard deviation: 23m 40s, 95% done after 1:09

Cross checks…






```
#ifdef GOOD_OLD_AOD
      AliAODTrack* track = (AliAODTrack*) event->GetTrack(i);
#else
      AliAODSpecialTrack* track = (AliAODSpecialTrack*) event->GetTrack(i);
#endif
```

(will not always be this trivial: derived custom vars
#ifdef may not be needed, see later)

Filtering

Filtering is similar to muons delta

SpecialReplicator

Selects tracks
Creates special tracks
It's used to filter branches on an extension

```
AliAODExtension* ext = aodH->AddFilteredAOD(aodfilename,title);
```

```
AliAODSpecialReplicator * murep = new AliAODSpecialReplicator("SpecialReplicator",
                                                              "remove non interesting tracks, "
                                                              "writes special tracks array tracks",
                                                              new AliAnalysisSpecialTrackCuts,
                                                              new AliAnalysisPrimaryVertices,
                                                              fMCMode);
```

```
ext->FilterBranch("tracks",murep);
ext->FilterBranch("vertices",murep);
```