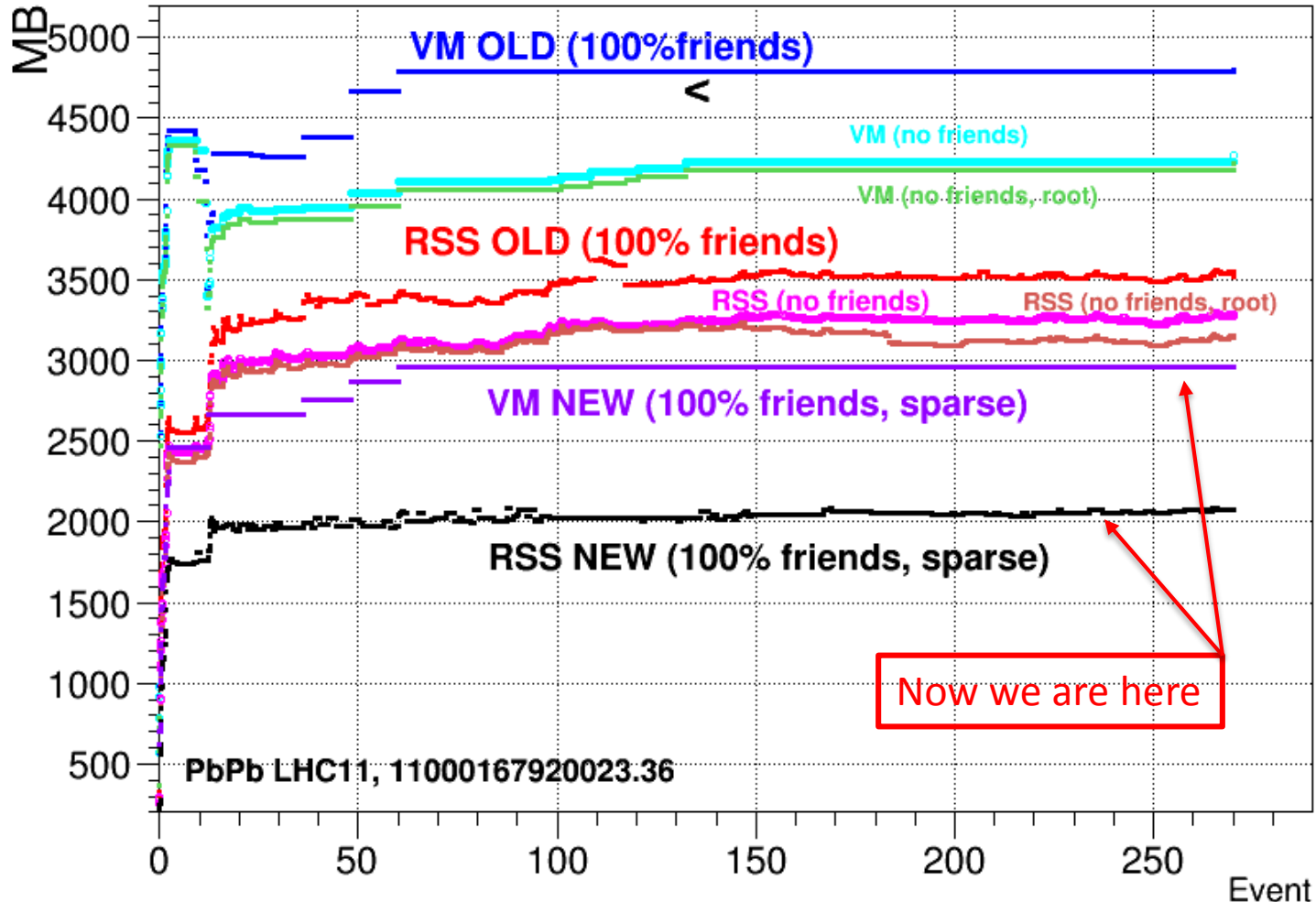


Reconstruction in Run 2 status and plans

Memory consumption

Tests on the heavy LHC11h chunk



Last results obtained with modified code (next slide) + moving TPC RecPoints creation to separate aliroot session before rec.C execution (Peter)

“OLD” means situation in September 2015

Changes for memory reduction ([ALIROOT-6411](#))

- ❑ Friend Tracks preparation modified:
 - No cloning of TPC calibration objects/friends: only pointers are reassigned
 - Friends for all tracks were stored: huge overhead from unused friend tracks (e.g. pile-up): now by default at most 3000 tracks will have friends, priority given to:
 - ITS & TPC & OuterDetectors
 - ITS & TPC & {TRD || TOF}
 - ITS & TPC
 - ITS_SA
 - If AliESDfriend prescaling is requested:
 - Decide in advance if friends for given event will be stored and fill it
 - For events w/o friends used dummy ESDfriend, don't delete main one
- ❑ Many optimizations in ITS, TPC, TRD code:
 - TPC seed size reduced (8.4kB → 3.5kB), seeds pool usage extended to KinkFinder too.
 - Cluster containers, pools for large repetitive memory allocations
 - Reduction of memory thrashing by allocating on stack rather than on the heap (option to increase aliroot stack)
- ❑ HLT memory optimization (Mattias)
- ❑ Discard TPC clusters from apparent pile-up (wrong CE side with 3 cm margin)

Prospects for further memory reduction

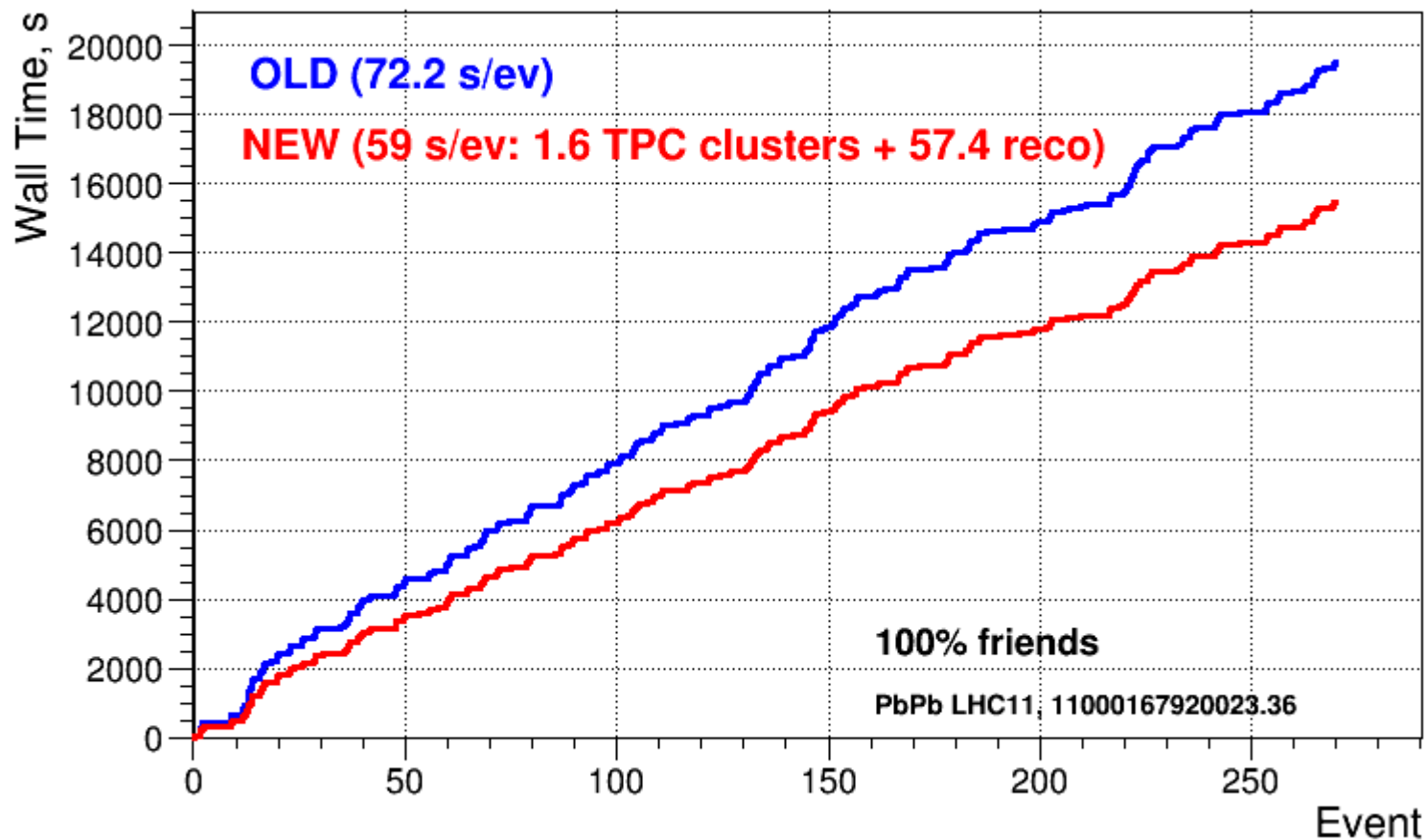
- ❑ Most of hotspots in TPC:
 - cluster containers use ~200MB just on pointers + navigation indices
~ 100 MB can be recovered
 - CalibDB: preloads ~200MB of OCDB objects, regardless on their need (the same feature shared by TRD)
⇒ possibility of objects size reduction / selective uploading should be studied by detector experts

- ❑ Multiple up to 10-20 MB savings possible with difference of code reshuffling

Worth to spend time on this only if current memory consumption appears still to be too high.

CPU consumption

Tests on the same heavy LHC11h chunk



Gain of ~20% as by-product of memory optimization
+ substitution of TPC “FastMath” by standard one (Giulio)

CPU consumption

Example of 400kHz LHC15g data

Incl.	Self	Called	Function	Location
100.00	0.00		(0) 0x00000000000012d0	ld-2.19.so
99.99	9.85	14 954	<cycle 16>	libCore.so.5.34
99.99	0.00	1	0x0000000000004030da	allroot
99.99	0.00	1	(below main)	libc-2.19.so: libc-start.c
99.99	0.00	1	main	allroot: allroot.cxx, TOb
99.99	0.00	1	TRint::Run(bool)	libRint.so.5.34: TRint.o
20.14	0.18	41 525 512	AliTrackerBase::MeanMaterialBudget(dou...	libESD.so: AliTrackerBa
18.25	2.01	106 858 445	AliTPCClusterParam::QlotCorrection(int, ...	libTPCbase.so: AliTPCC
17.34	0.01	108 063 004	TGeoManager::FindNextBoundaryAndStep...	libGeom.so.5.34: TGeof
17.31	0.16	108 063 004	TGeoNavigator::FindNextBoundaryAndSte...	libGeom.so.5.34: TGeof
16.39	6.63	6 796 449 351	AliTPCClusterParam::GaussConvolution(d...	libTPCbase.so: AliTPCC
15.07	0.17	65 176 142	AliTPCTransform::Transform(double*, int*, ...	libTPCbase.so: AliTPCT
14.79	0.04	130 352 284	AliTPCCorrection::CorrectPoint(float*, short)	libTPCbase.so: AliTPCC
14.75	0.46	130 352 284	AliTPCComposedCorrection::GetCorrectio...	libTPCbase.so: AliTPCC
12.56	0.21	108 061 693	TGeoNavigator::FindNextDaughterBounda...	libGeom.so.5.34: TGeof
10.63	0.83	3 662 260 384	sincos	libm-2.19.so: s_sincos.c
9.63	0.20	142 486 645	TGeoShapeAssembly::DistFromOutside/do...	libGeom.so.5.34: TGeof

Hot spots for high-multiplicity data

~20% : material budget query via Tgeo.

Should be substituted (at least in all queries except for final fits) by fast lookup tables. Ultimately, if we succeed to use LUTs everywhere, we don't need to keep geometry in the memory but to only fetch the matrices.

Development relevant also for O2

CPU consumption

Example of 400kHz LHC15g data

Incl.	Self	Called	Function	Location
100.00	0.00		(0)	ld-2.19.so
99.99	9.85	14 954	<cycle 16>	libCore.so.5.34
99.99	0.00	1	0x00000000004030da	allroot
99.99	0.00	1	(below main)	libc-2.19.so: libc-start.c
99.99	0.00	1	main	allroot: allroot.cxx, TOT
99.98	0.00	1	TRint::Run(bool)	libRint.so.5.34: TRint.c
20.14	0.10	41 525 512	AliTrackerBase::MeanMaterialBudget(dou...	libESD.so: AliTrackerBa
18.25	2.01	106 850 445	AliTPCClusterParam::QtotCorrection(int, l...	libTPCbase.so: AliTPCC
17.34	0.01	108 003 004	TGeoManager::FindNextBoundaryAndStep...	libGeom.so.5.34: TGeof
17.31	0.16	108 063 004	TGeoNavigator::FindNextBoundaryAndSte...	libGeom.so.5.34: TGeof
16.39	6.63	6 796 449 351	AliTPCClusterParam::GaussConvolution(d...	libTPCbase.so: AliTPCC
15.07	0.17	65 176 142	AliTPCTransform::Transform(double*, int*, ...	libTPCbase.so: AliTPCT
14.79	0.04	130 352 284	AliTPCCorrection::CorrectPoint(float*, short)	libTPCbase.so: AliTPCC
14.75	0.46	130 352 284	AliTPCComposedCorrection::GetCorrectio...	libTPCbase.so: AliTPCC
12.56	0.21	108 061 693	TGeoNavigator::FindNextDaughterBounda...	libGeom.so.5.34: TGeof
10.63	0.83	3 662 260 384	sincos	libm-2.19.so: s_sincos.c
9.63	0.20	142 486 645	TGeoShapeAssembly::DistFromOutside/do...	libGeom.so.5.34: TGeof

Hot spots for high-multiplicity data

~18% : TPC dE/dX evaluations: multidimensional Gaussian convolutions.

Can we consider faster, even if less precise algorithm? To be discussed with TPC

CPU consumption

Example of 400kHz LHC15g data

Incl.	Self	Called	Function	Location
100.00	0.00		(0)	ld-2.19.so
99.99	9.85	14 954	<cycle 16>	libCore.so.5.34
99.99	0.00			allroot
99.99	0.00		(below main)	libc-2.19.so: libc-start.c
99.99	0.00		main	allroot: allroot.cxx, TOT
99.98	0.00		TRint::Run(bool)	libRint.so.5.34: TRint.c
20.14	0.18	41 525 512	AliTrackerBase::MeanMaterialBudget(dou...	libESD.so: AliTrackerBa
18.25	2.01	106 850 445	AliTPCClusterParam::QtotCorrection(int, l...	libTPCbase.so: AliTPCC
17.34	0.01	108 063 004	TGeoManager::FindNextBoundaryAndStep...	libGeom.so.5.34: TGeof
17.31	0.16	108 063 004	TGeoNavigator::FindNextBoundaryAndSte...	libGeom.so.5.34: TGeof
15.07	0.17	65 176 142	AliTPCTransform::Transform(double*, int*, ...	libTPCbase.so: AliTPCT
14.79	0.04	130 352 284	AliTPCCorrection::CorrectPoint(float*, short)	libTPCbase.so: AliTPCC
14.75	0.46	130 352 284	AliTPCComposedCorrection::GetCorrectio...	libTPCbase.so: AliTPCC
12.56	0.21	108 061 693	TGeoNavigator::FindNextDaughterBounda...	libGeom.so.5.34: TGeof
10.63	0.83	3 662 260 384	sincos	libm-2.19.so: s_sincos.c
9.63	0.20	142 486 645	TGeoShapeAssembly::DistFromOutside(do...	libGeom.so.5.34: TGeoS

Hot spots for high-multiplicity data

~15% : cluster raw to coordinates conversion + corrections

Currently AliTPCComposedCorrection is used: ~6 μ s/cluster







Once the new distortion correction framework is in place, one single correction via AliTPCChebCorr will be applied:

Speed for “standard” corrections (a la Run1, no “SCD”) < 1 μ s/cluster

With observed SCD speed depends on the smoothness of the underlying distortions, was ~2 μ s/cluster for not particularly smooth map.

CPU consumption

Example of 400kHz LHC15g data

4.26	3.94	3 736 375 621		__sin_avx	libm-2.19.so: s_sin.c, fe
4.18	1.05	1 364 708 498		TGeoXtru::DistToPlane(double const*, dou...	libGeom.so.5.34: TGeo
3.97	1.12	325 880 710		AliTPCCalibGlobalMisalignment::GetCorre...	libTPCbase.so: AliTPCC
3.85	0.09	193 917 400		AliMagF::Field(double const*, double*)	libSTEERBase.so: AliMa
3.84	3.59	130 352 284		AliMagWrapCheb::GetTPCRatIntCyl(doubl...	libSTEERBase.so: AliMa
3.76	2.65	193 916 574		AliMagWrapCheb::Field(double const*, do...	libSTEERBase.so: AliMa

Hot spots for high-multiplicity data

~8% : B-field and its integrals (for TPC) calculation

- For ITS constant field should be used (and was at some time)
- Outside of ITS, for preliminary navigation (seeding, rough extrapolations) fast HLT-like parameterization (R-dependent) should be used: corresponding switch will be added to AliMagF
- 3D B-field calls are largely abused: there is no point in propagating on short distances with 3D field even for the final fits
⇒ partially fixed in feature-devcalib branch

TPC distortions

On the description / extraction of SC distortions see Marian's slides on Tuesday: [1](#), [2](#)

Framework for distortions correction:

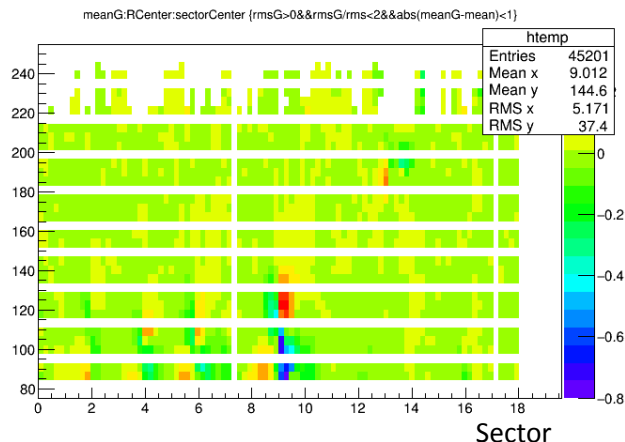
The final distortions maps for time-slices of ~30 min will be packed to Chebyshev polynomials (AliTPCChebCorr) parameterizations as OCDB objects (~2-3MB per slice, if maps provided from CPass0 are smooth)

TPC tracking code was modified to be able to correct for large distortions

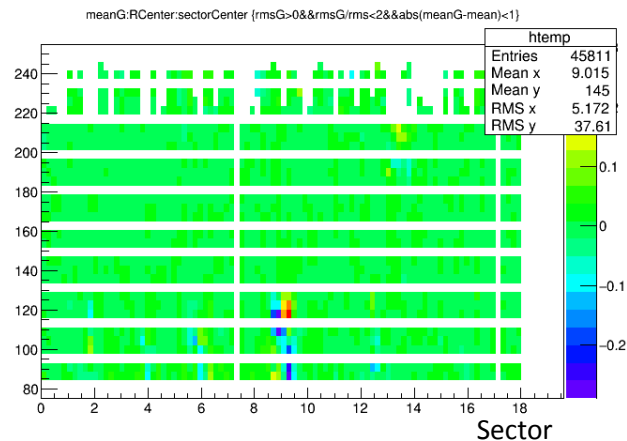
- Override all corrections by single AliTPCChebCorr parameterization
- Mechanism for uploading time-dependent correction maps and interpolating between pair of nearest time-slices
- Evaluate cluster (seeding, fits) not at the nominal pad-row X but at real corrected X
- Account for dead zones modifications due to the distortions

Code is in feature-devcalib, once the maps are ready, will be merged to master

Before
correction



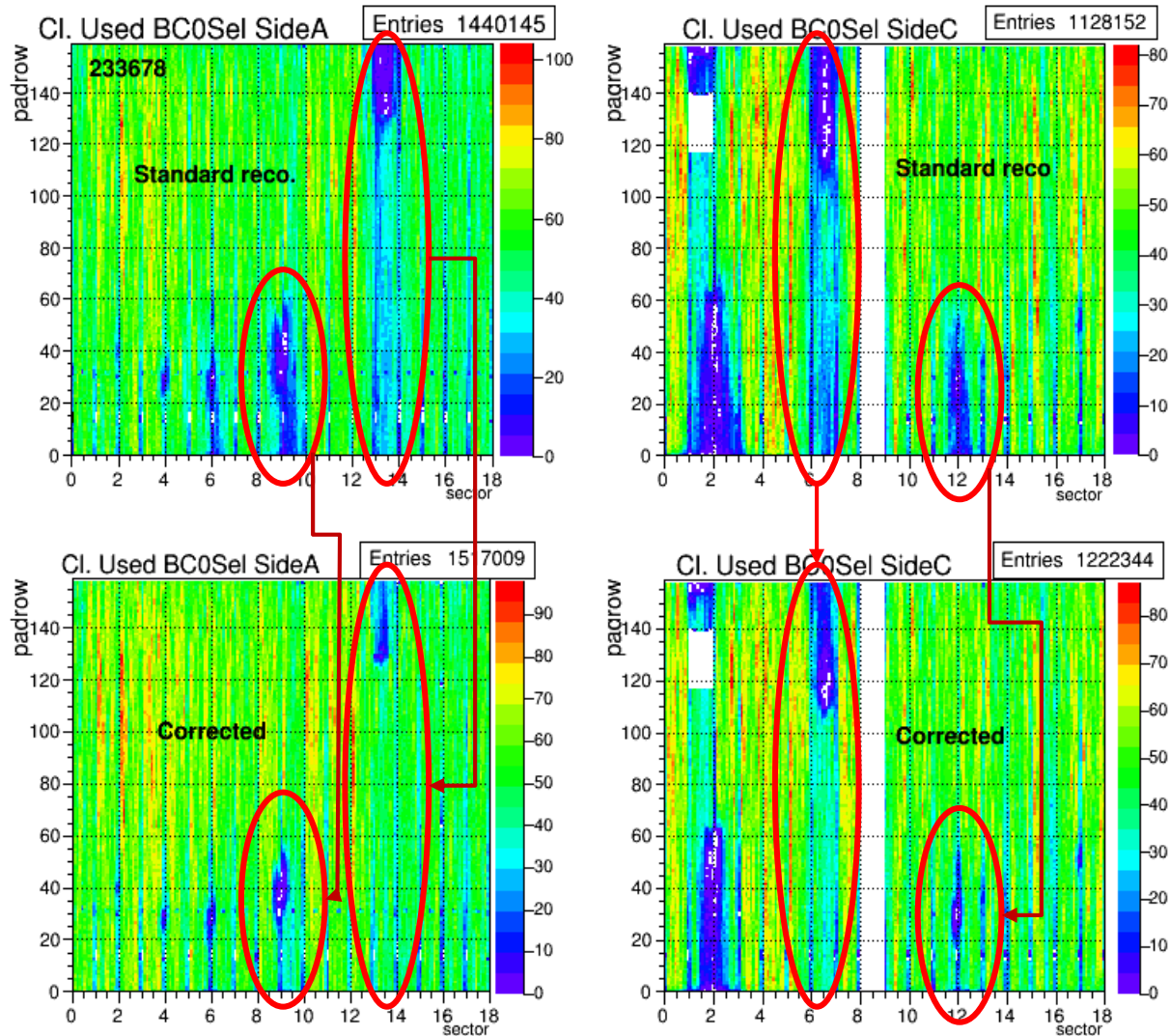
R



After
correction
(incomplete)

TPC cluster usage by non-pile-up (TOF BC=0 selected) tracks

Not all holes closed (e.g. clusters recovered): reconstruction for distortions extraction used insufficiently relaxed tracking tolerances (fixed)

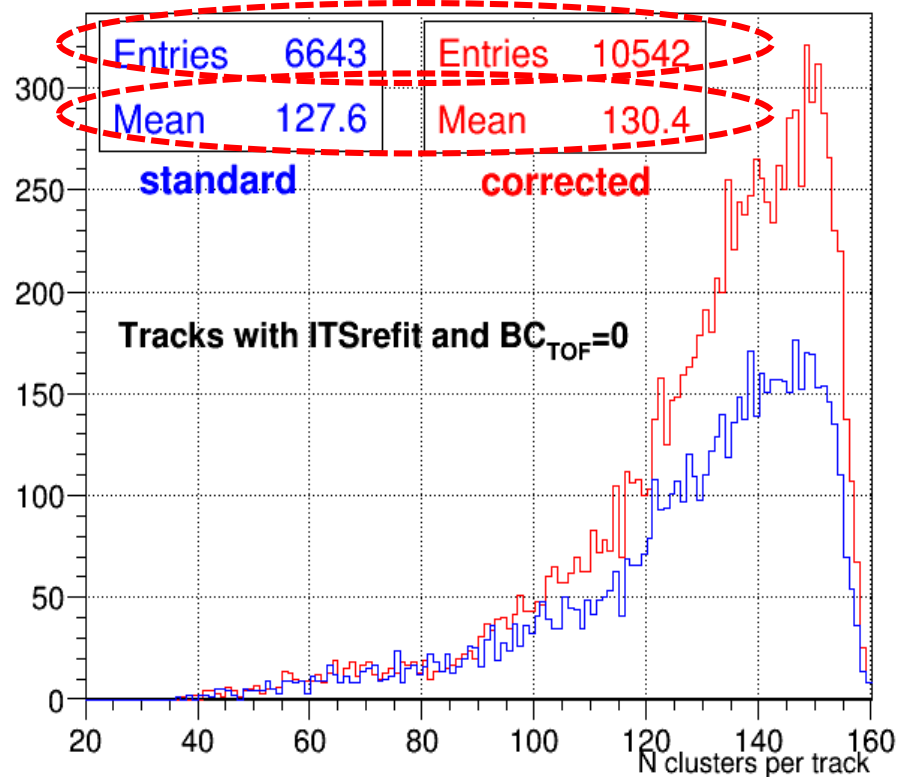


Test on 400 kHz IR run 233678

Correction framework works: comparing standard reconstruction with reconstruction using new correction:

ITS Matching rate for triggered BC increases by ~60%

N TPC clusters per track also increases



Note: ITS-matching was affected not only by the TPC track loss due to the distortions but also by wrong extrapolation from TPC to ITS

Distortions lead to cluster losses in dead zones between the sectors

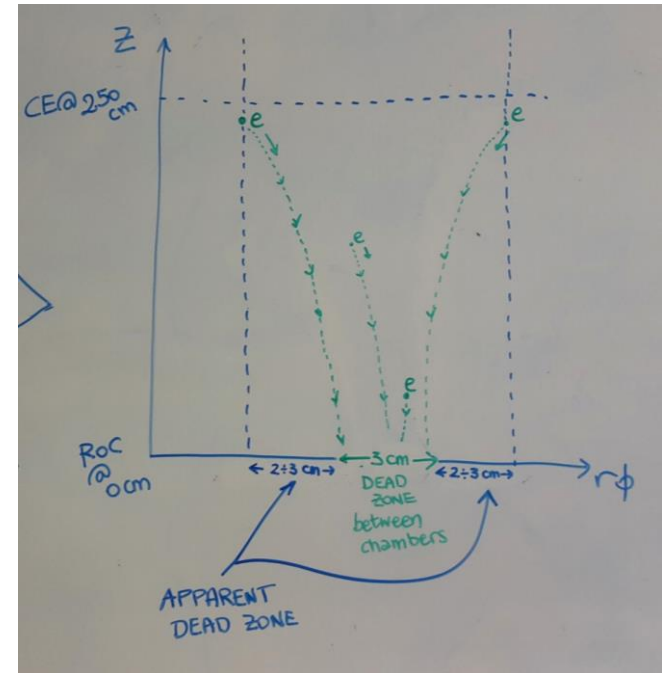
⇒ distorting in MC and correcting in reconstruction is NOT equivalent to recovering “no-distortions” scenario (even modulo residual mis-calibrations)

Proper cuts should be applied in the analysis to account for modifications of dead zones.

Framework is prepared (in feature-devcalib)

(but not yet fully tested due to the absence of the maps)

- AliTPCChEbDist (~ inverse of AliTPCChEbCorr) will override all distortions applied currently in the MC
- MC events are simulated with the real time stamp from GRP, every MC chunk is generated with ordered time-stamps covering full run range
- The luminosity decay at the moment is accounted by exponential dumping of event probability vs time (decay time provided as parameter, eventually can be extracted from OCDB)



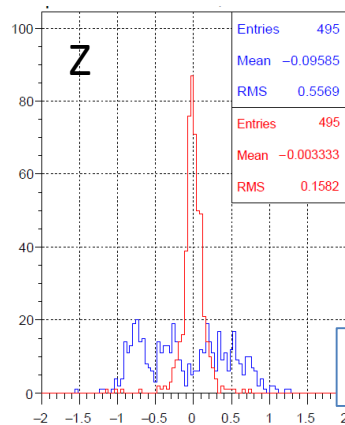
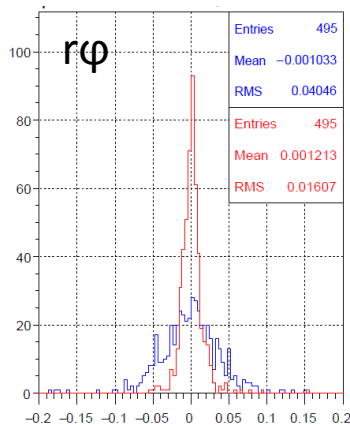
Including TRD into the tracking

Task was paused after realizing that the showstopper is the TPC calibration (even w/o SC distortions...)

The prerequisite (both for TRD in tracking and TPC calibration):

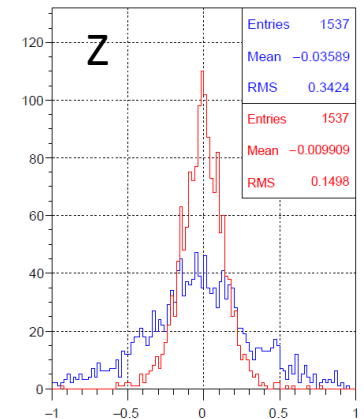
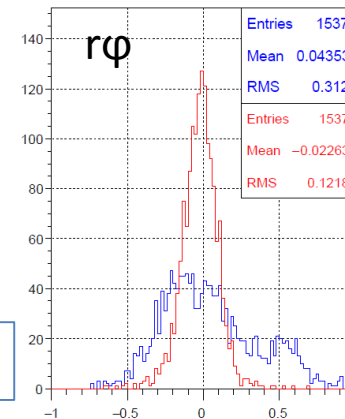
<200 μ m global alignment of ITS, TRD and TOF w/o using TPC in the fits is fulfilled

TRD <residuals> per chamber: **before**, **after** alignment

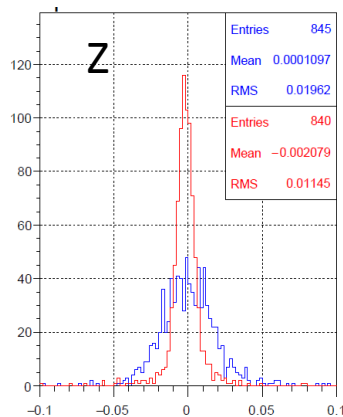
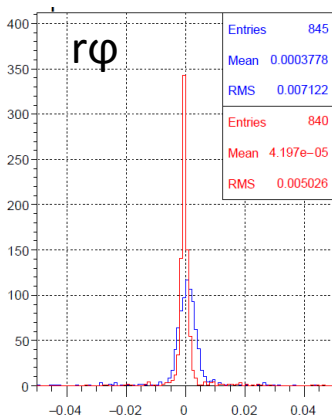


LHC15f

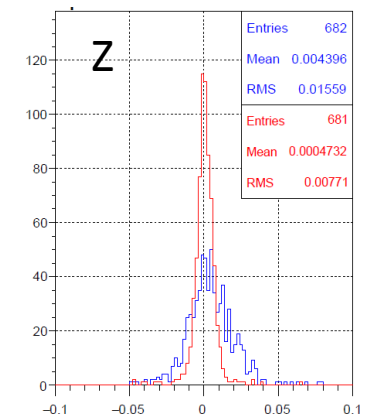
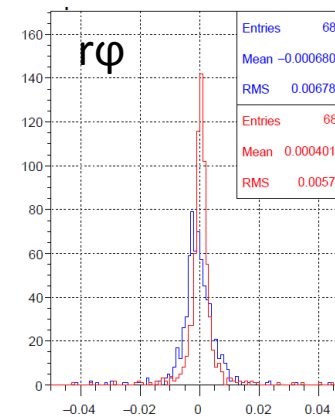
TOF <residuals> per strip: **before**, **after** alignment



ITS Lr6 residuals per module: **before**, **after** alignment



ITS Lr5 residuals per module: **before**, **after** alignment



Including TRD into the tracking

Still to do (after solving TPC distortions problem):

- TRD ExB calibration is currently tuned on the MC with ad-hoc factors
- The position and error assignment for pad-row crossing TRD tracklets is unsatisfactory (wrong?)
- The tracklet reconstruction part of the code is unmanagable

While including TRD with current status in tracking already will improve the pT resolution, to achieve the best result the data-driven calibration coupled to alignment should be performed within Millepede framework.

⇒ need an interface for TRD calibration TOF for Millepede

On global scope:

In Run3 TRD will use directly online-tracklets for reconstruction

The work on corresponding tracker started but at the moment is suspended

⇒ the aim should be to rewrite current TRD tracker in such a way that it serves as prototype for O2 and still can be used in Run2

Currently we use:

- in Pb-Pb: “old” vertexer (only 1 “primary” vertex is reconstructed)
- in p-p and pA: “multi-vertexer” (able to reconstruct also pile-up)

Both are relying on the same fitting routine, slow and not well adapted to Alice track model

Implemented for the HLT ITS-SA tracking/vertexing component:

- more efficient and precise vertex fitter
- the logics of outliers rejection in vertex search improved (though no search for multiple vertices is performed in the HLT))

This should evolve to new vertexer with better pile-up tagging capabilities (Bayesian logics accounting for IR and BC filling)

