Atlas IO improvements and Future prospects

Wahid Bhimji, Ilija Vukotic

Most slides from Ilija….

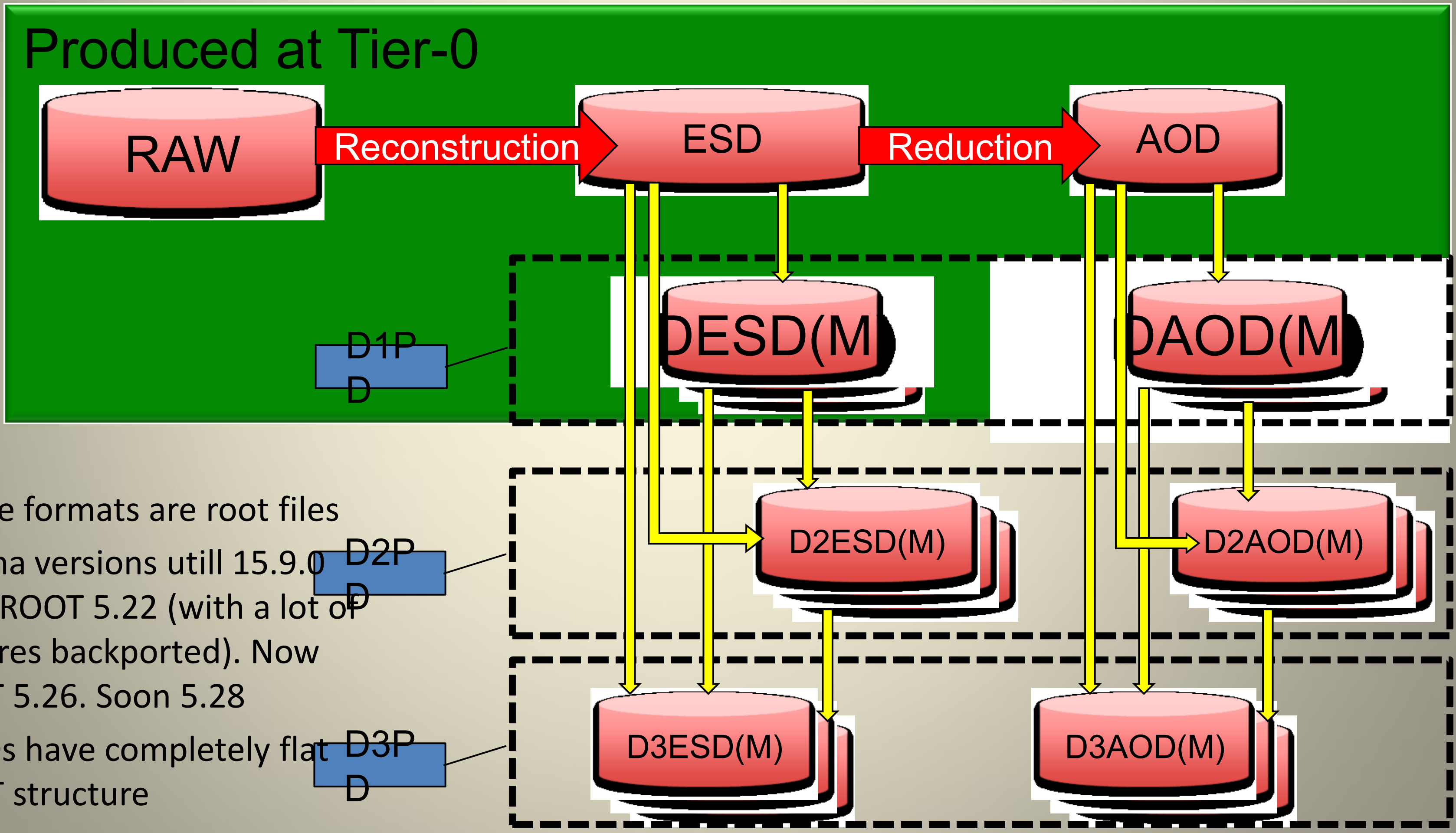
THE UNIVERSITY OF EDINBURGH

Overview: From past to future

- PAST (Up until 2010)
  - Unordered ROOT files – horrible I/O

    -> Basket Ordering / TTreeCache
- CURRENT
  - AutoFlush / New Root Versions
- FUTURE
  - Near
  - Far

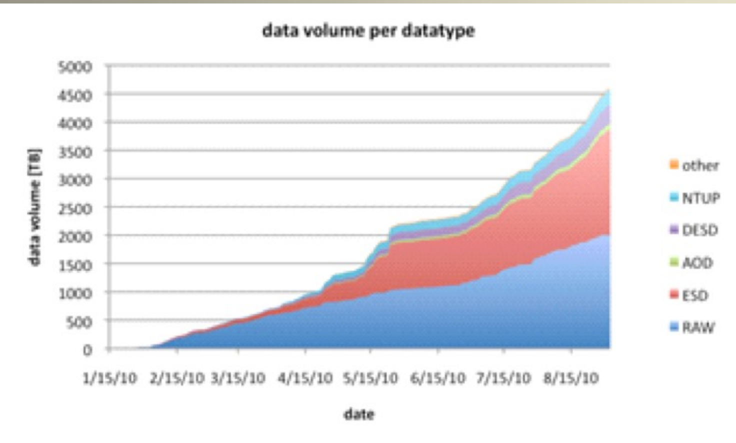
Formats



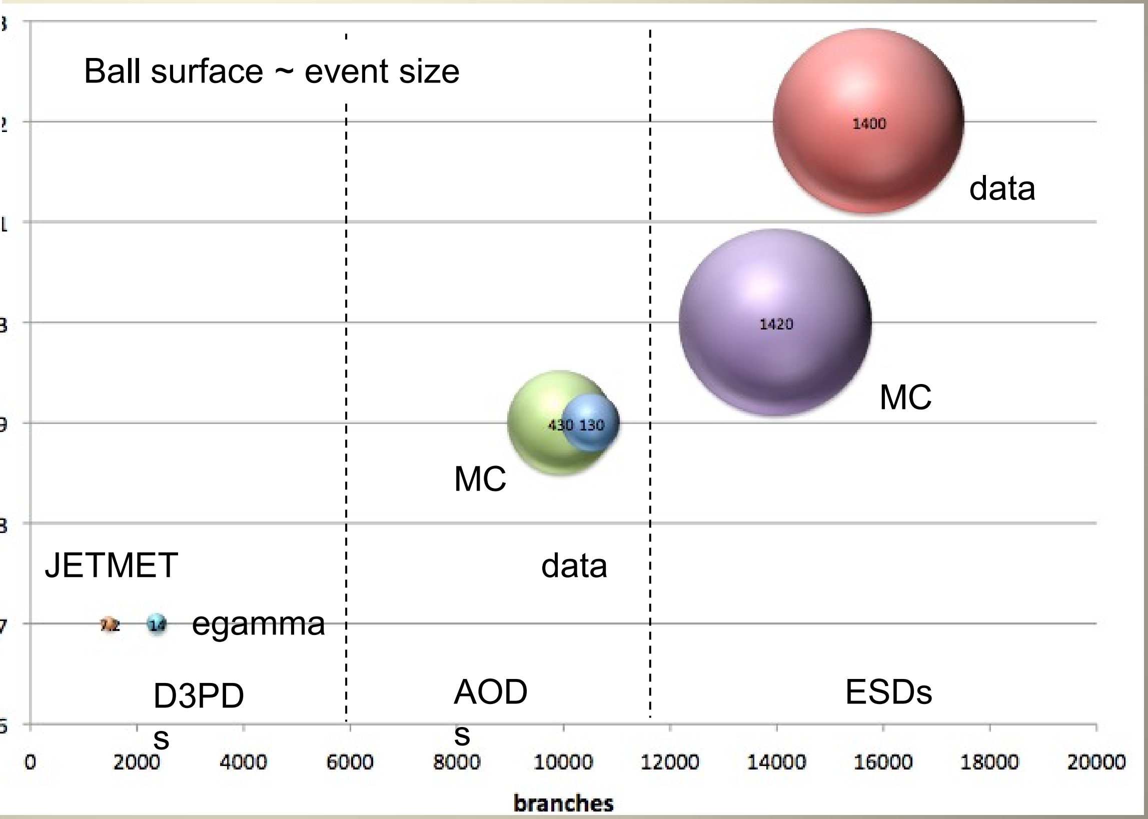
- All the formats are root files
- Athena versions utill 15.9.0 used ROOT 5.22 (with a lot of features backported). Now ROOT 5.26. Soon 5.28
- D3PDs have completely flat ROOT structure

Formats

At CHEP last year
– so not current but idea of scale

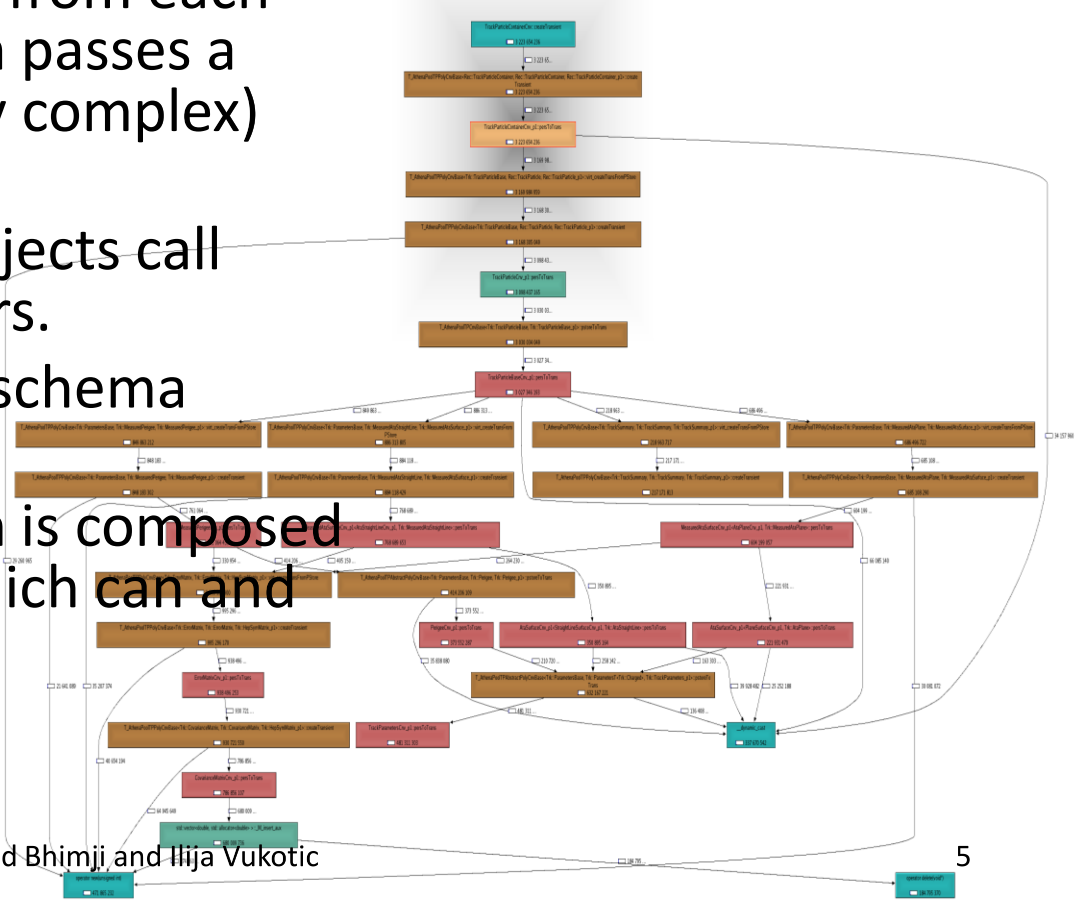


Format	Size [PB]
RAW	2
ESD	1.8
AOD	0.1
DESD	0.3
D3PD	0.3



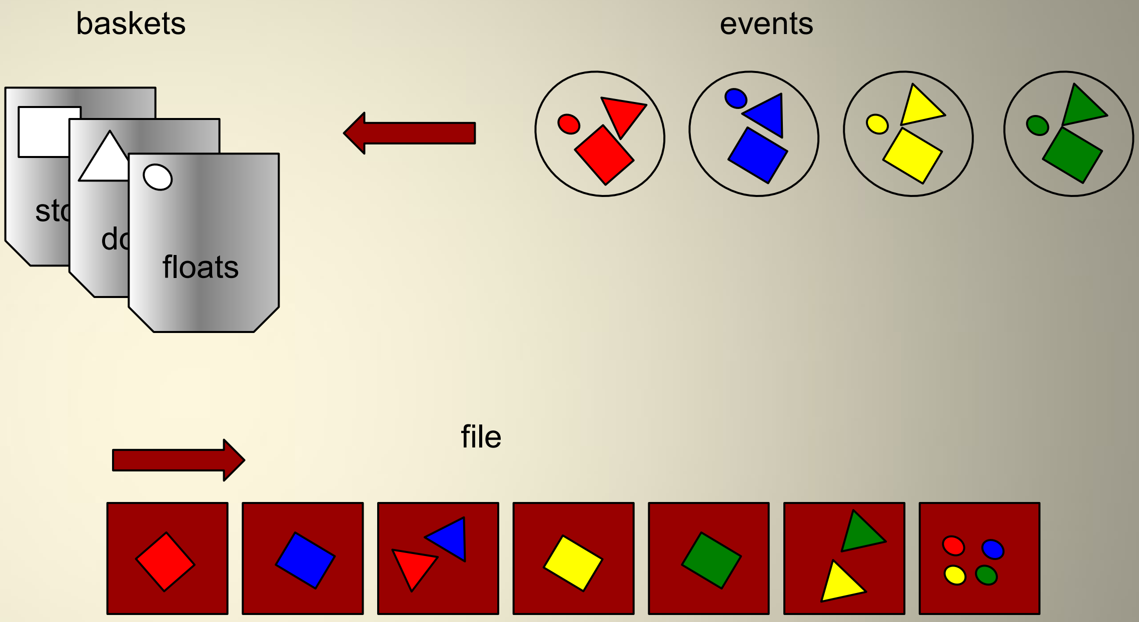
transient/persistent split

- Transient objects are converted to persistent ones.
- To store it efficiently data from each sub-detector or algorithm passes a different (sometimes very complex) set of transformations.
- Converters of complex objects call converters for its members.
- It provides possibility for schema evolution
- Example: TracksCollection is composed of 20 different objects which can and do evolve individually


TrackParticleContainerCnv::createTransient
3 223 654 236
3 223 65...
T_AthenaPoolTPPolyCnvBase<Rec::TrackParticleContainer, Rec::TrackParticleContainer, Rec::TrackParticleContainer_p1>::createTransient
3 223 654 236
3 223 65...
TrackParticleContainerCnv_p1::persToTrans
3 223 654 236
3 169 98...
T_AthenaPoolTPPolyCnvBase<Trk::TrackParticleBase, Rec::TrackParticle, Rec::TrackParticle_p1>::virt_createTransFromPStore
3 169 684 859
3 168 38...
T_AthenaPoolTPPolyCnvBase<Trk::TrackParticleBase, Rec::TrackParticle, Rec::TrackParticle_p1>::createTransient
3 168 385 048
3 098 43...
TrackParticleCnv_p1::persToTrans
3 098 437 165
3 030 03...
T_AthenaPoolTPCnvBase<Trk::TrackParticleBase, Trk::TrackParticleBase_p1>::pstoreToTrans
3 030 034 049
3 027 34...
TrackParticleBaseCnv_p1::persToTrans
3 027 346 393
886 313 ...
218 963 ...
886 313 805
218 963 717
884 118 ...
217 171 ...

Root File Organisation: Past -> Present

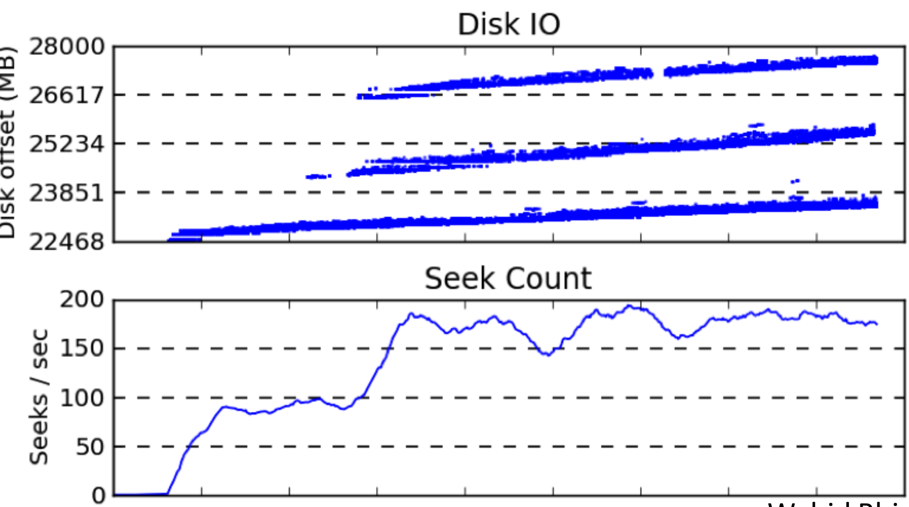
- Currently fully split (better compression)
- Baskets are written to file as soon as they get full - that makes parts of the same event scattered over the file.
- 2010: re-writing the files with baskets reordered.
- 2011: Autoflush

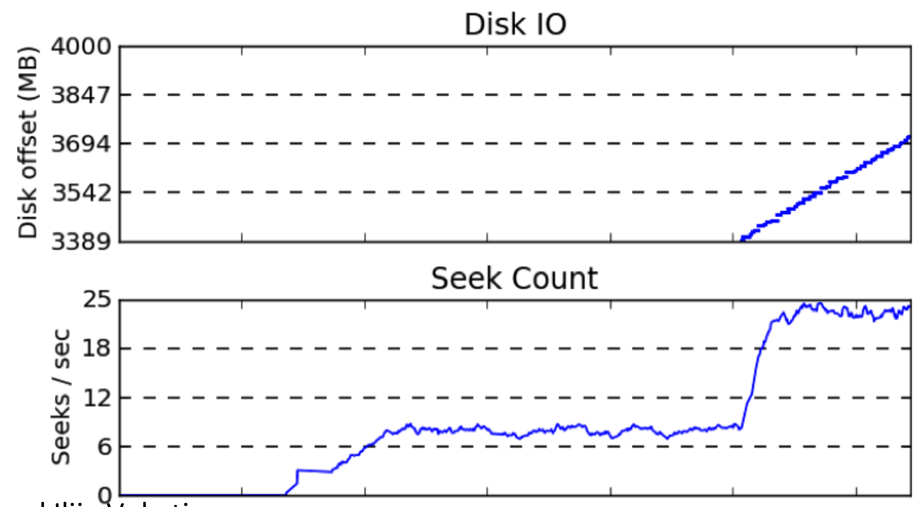


AutoFlush – after first *n* events to write are collected, baskets are resized in order to have the same number of baskets per branch. In that way all the branches should be “synchronized”.

The Past: Basket reordering improvements

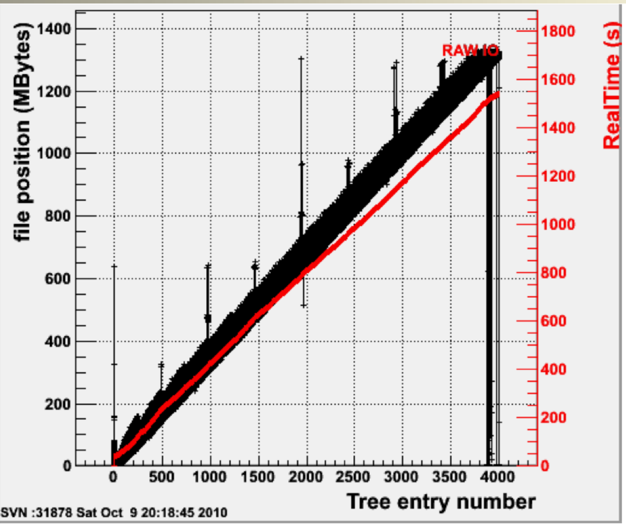
- Six ATLAS athena D3PDMaker jobs running on different 2GB AOD files accessing one disk partition.
- Unordered files show large scatter in reads and hit seek limits.
- Ordered files is significantly more linear and so seek counts reduced.



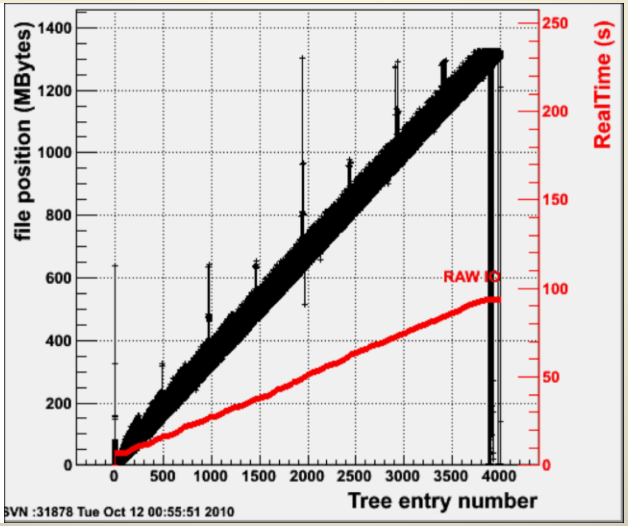


The past: Basket ordering - remote reading example

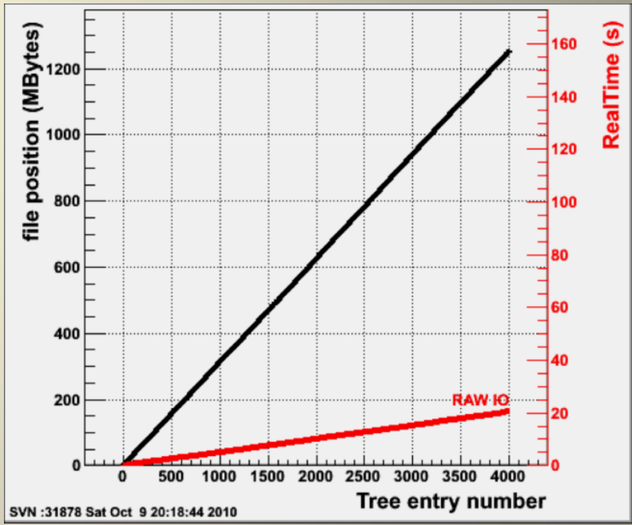
- ROOT test on these AODs, output from TTreePerfStats.
- GPFS running on same site as DPM.
- Ordering makes a much bigger impact.



Unordered - DPM (Rfio)
Disk Time 1500s
Wall Clock 1700s‘



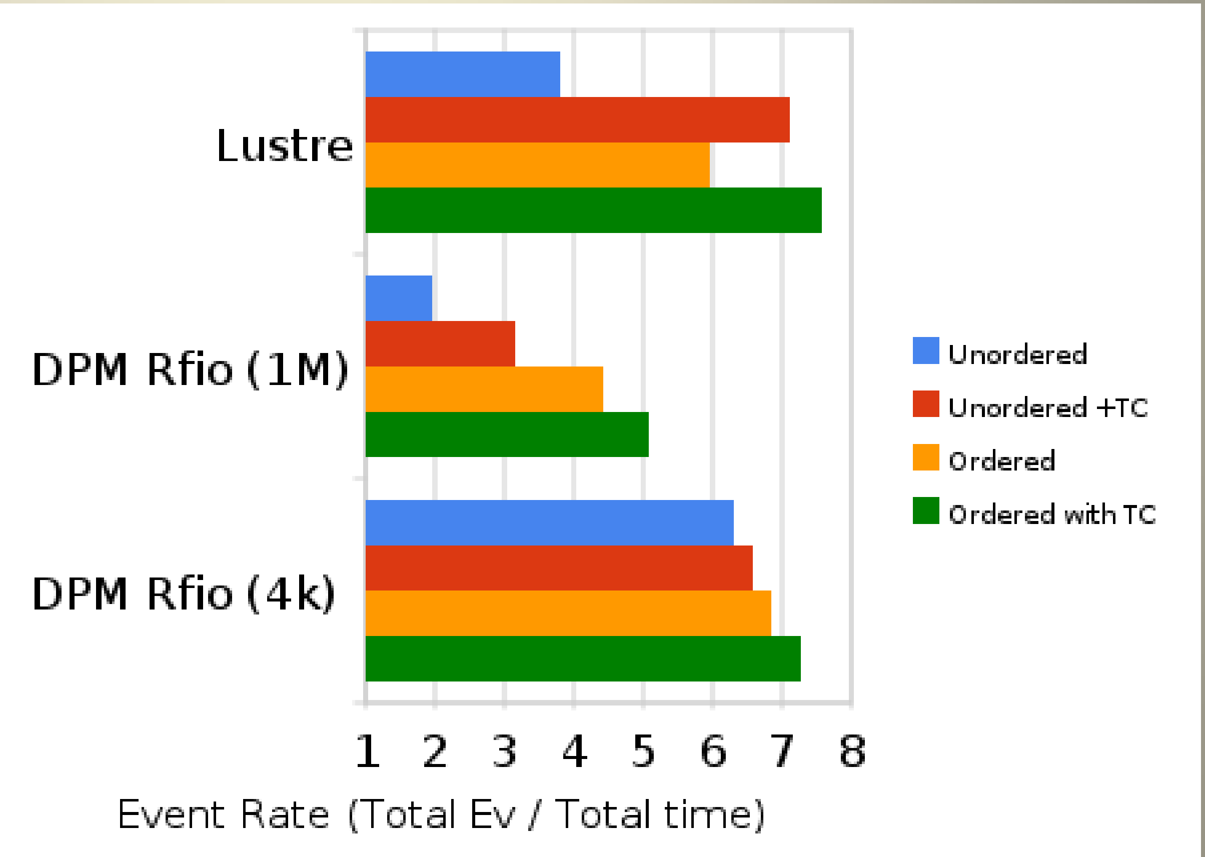
Unordered - GPFS
Disk time 100s
Wall Clock 230s



Ordered - DPM (Rfio)
Disk time 20s
Wall Clock 160s

The past: TTreeCache also showed an improvement …

- Varies by site storage type
- Still site settings like read-ahead buffers can make big difference



The present: Current running

Full split, **autoflush at 30MB** for the largest tree.
Other trees with fixed basket size.

ROOT 5.26/00e

Format	branches	Event size	Zip		mem
			level	factor	
ESD	12799	1272.34	6	3.40	34.90
AOD	8231	177.73	6	3.38	33.44
DPD - EGAMMA	2500	23.96	1	2.86	41.09
DPD – PHOTON	1490	13.94	1	2.53	28.57
DPD – JETMET	5671	90.12	1	2.88	28.39
DPD - SUSY	2132	14.95	1	3.65	28.61

The very near future (17.0.1)

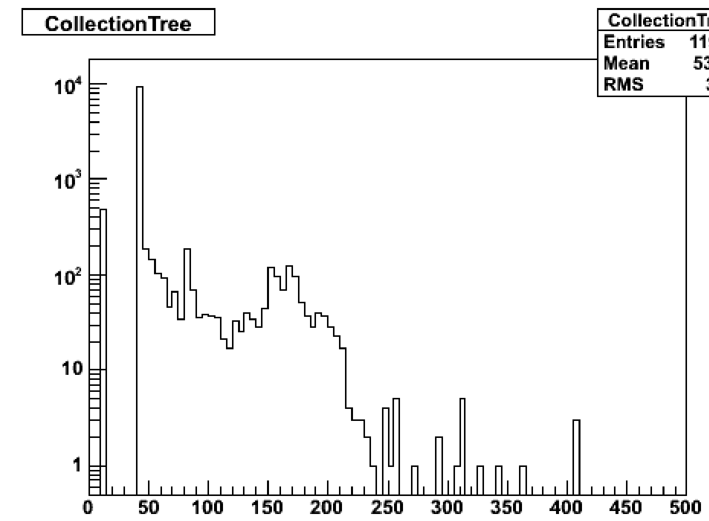
- **Reduced buffer size**
  To improve read time when sparse read of events we plan to change default (30MB) to memory equivalent of 10 events (AOD) and 5 events (RDO,ESD)
- **Split 0**
  Drastically reduces number of baskets – but not to 300ish*
- **Memberwise streaming**
- **Reduce of zip level**
  Current optimums are 4 (ESD) and 5 (AOD)

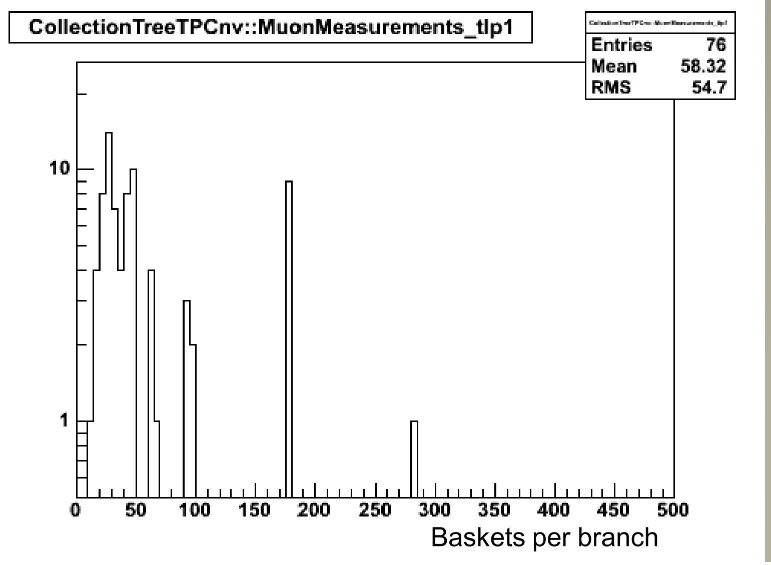
ROOT 5.28/00b

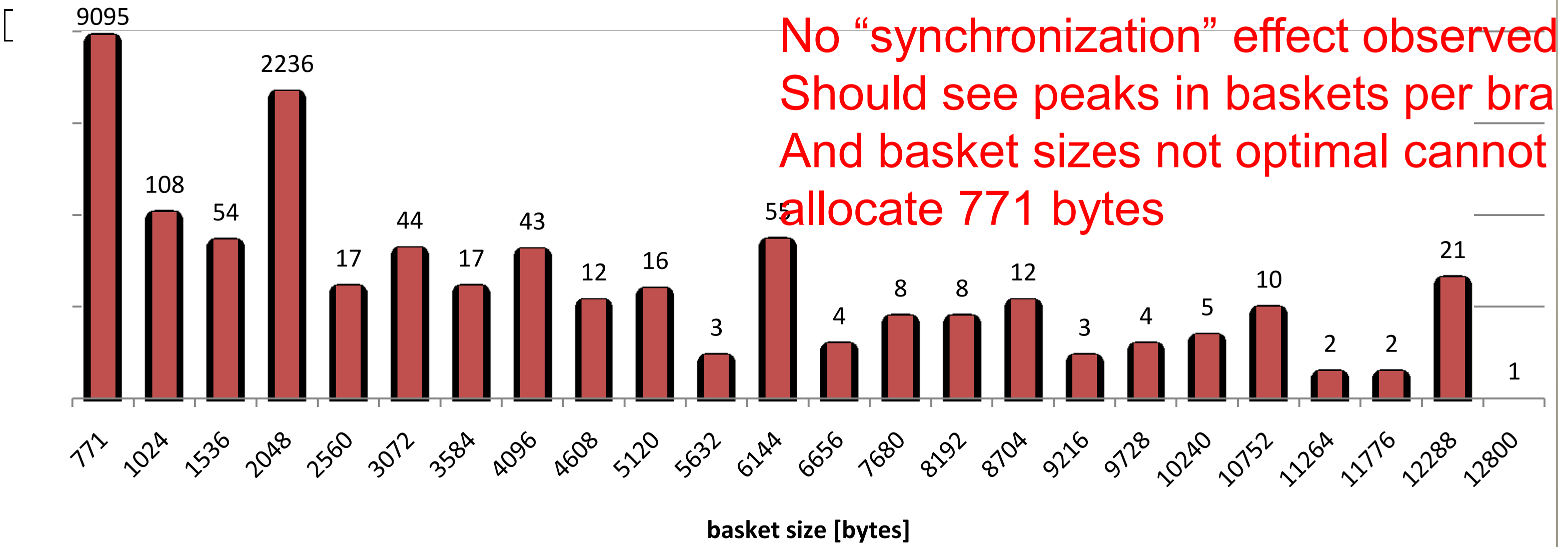
Format	branches	Event size [kb]	Zip		Mem [MB]
			level	factor	
ESD	807*	1338.63 ⬆ 8%	3	3.56	19.88
AOD	658*	191.67 ⬆ 5%	3	3.48	5.52

* Two very large and fast containers remain fully split for size benefits.

ESD







No "synchronization" effect observed!
Should see peaks in baskets per branch
And basket sizes not optimal cannot e.g. allocate 771 bytes

Athena reading – CPU only

Format	Root read speed [MB/s]	Athena reading		
		Full* [MB/s]	Speed [MB/s]	Time [ms]
ESD	13.05	3.41	5.93	192.45
ESD 17.0.1	19.71	3.85	7.27	161.00
AOD	4.57	2.85	3.15	43.00
AOD 17.0.1	10.32	4.91	6.42	25.86

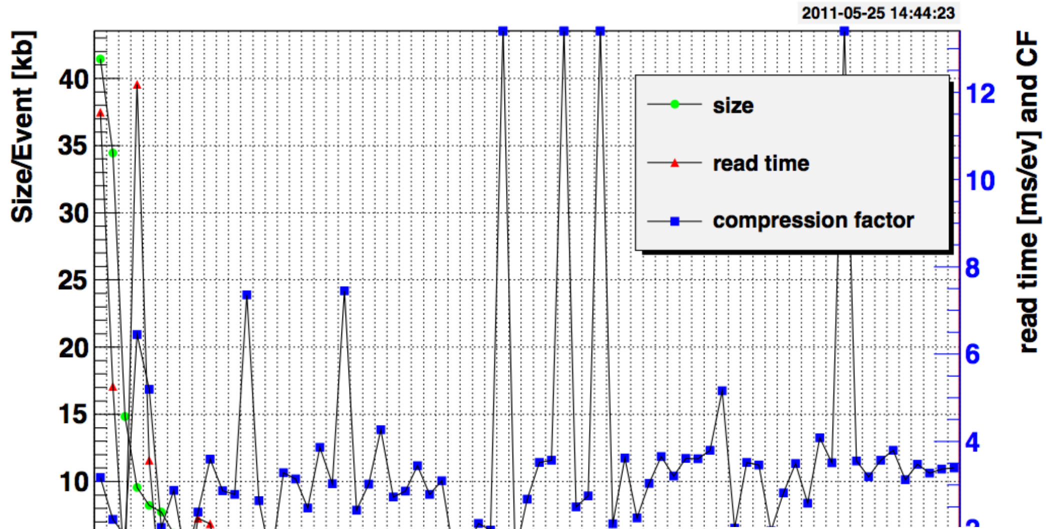
Two effects folded

- Improvements in TP converters
- New ROOT version

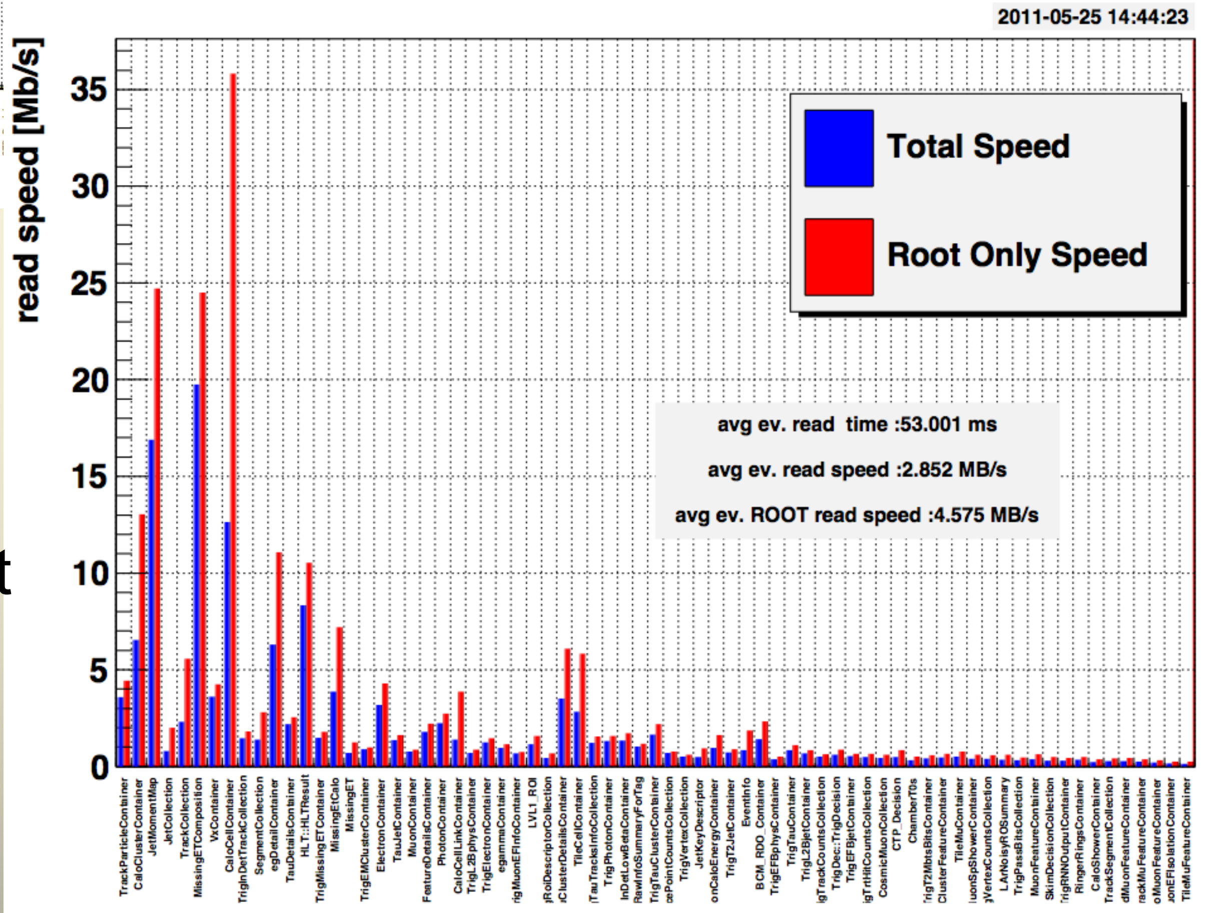
* Includes time to recreate certain objects (ie. CaloTowers)

Present

AOD



X-axis:
Collections ordered in size, so left are largest

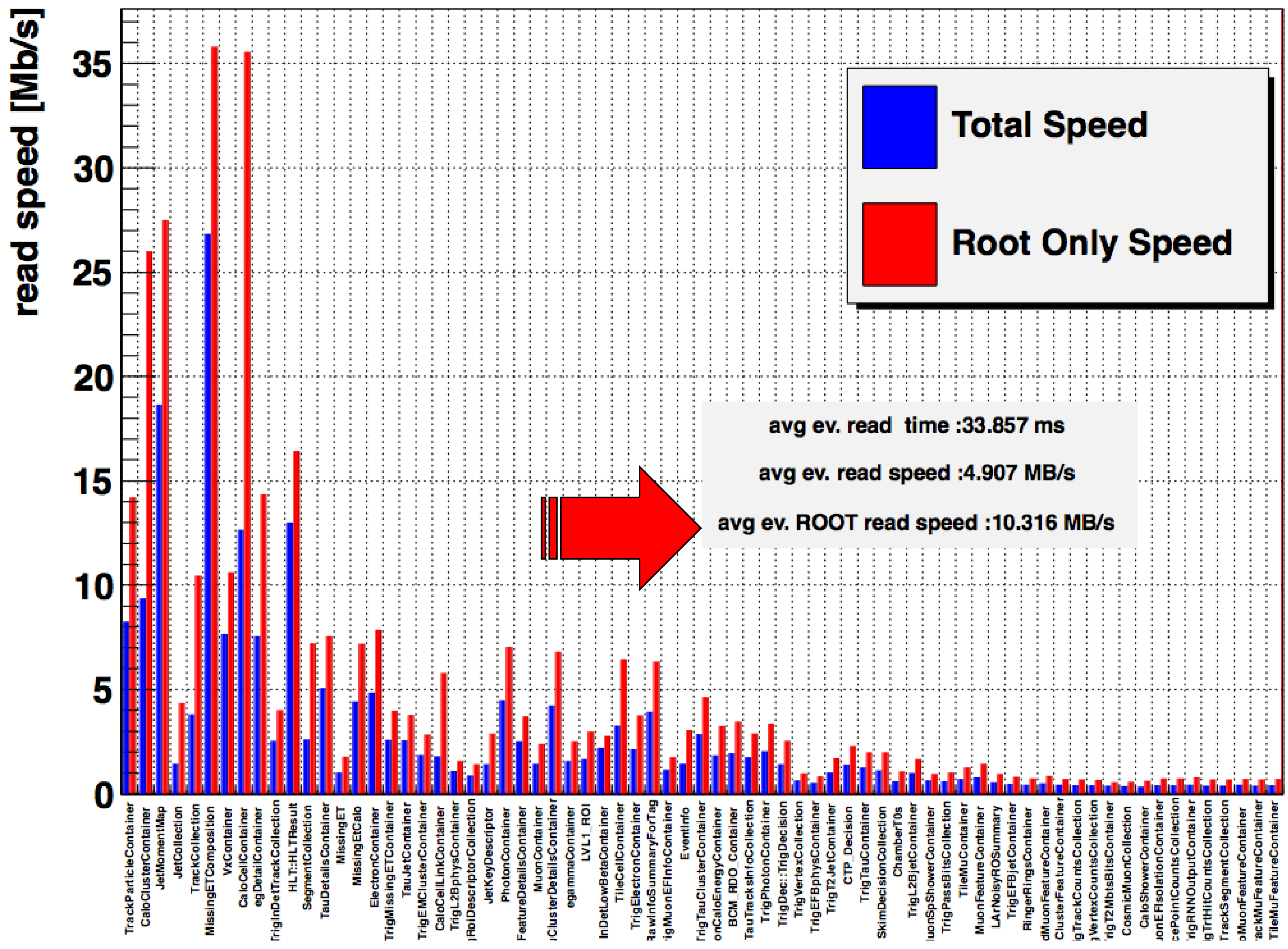


avg ev. read time :53.001 ms

avg ev. read speed :2.852 MB/s

avg ev. ROOT read speed :4.575 MB/s

Near Future: AOD



avg ev. read time :33.857 ms

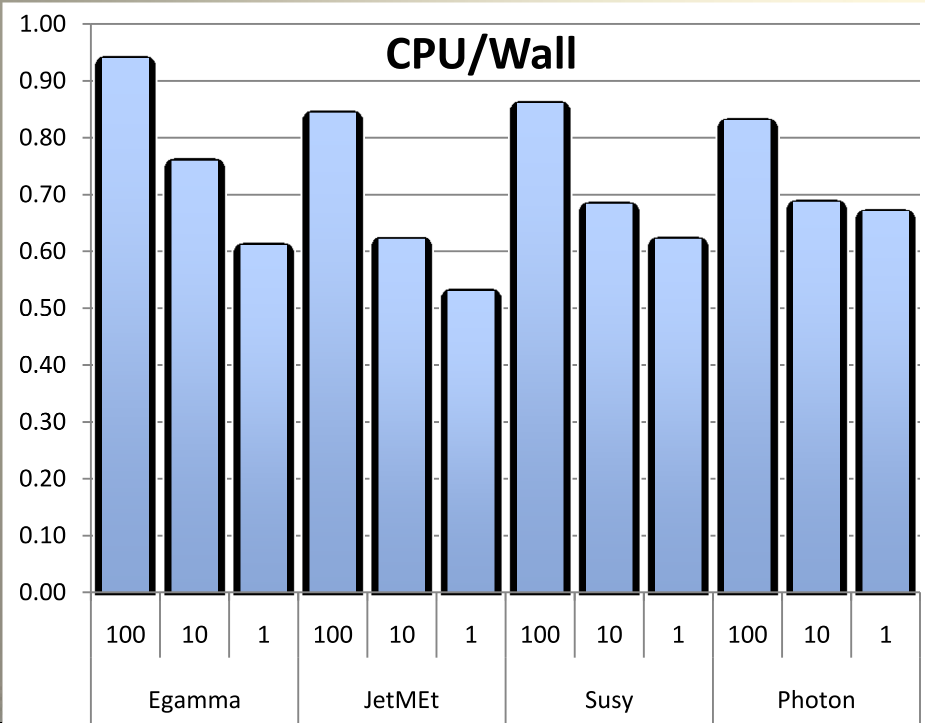
avg ev. read speed :4.907 MB/s

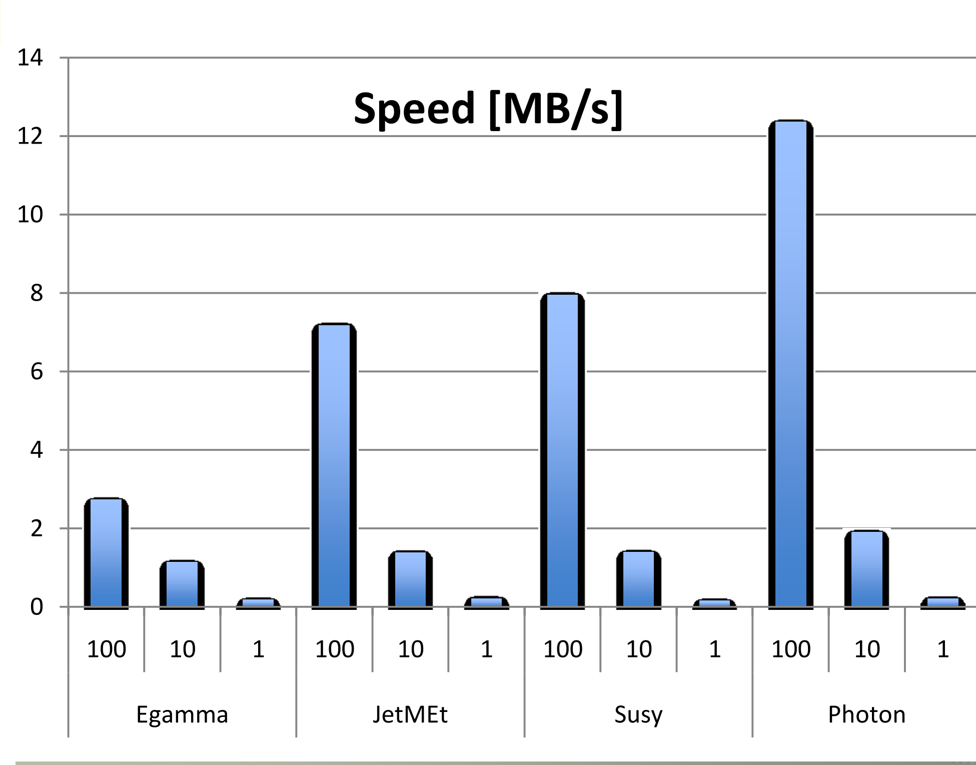
avg ev. ROOT read speed :10.316 MB/s

D3PD

WALL CLOCK

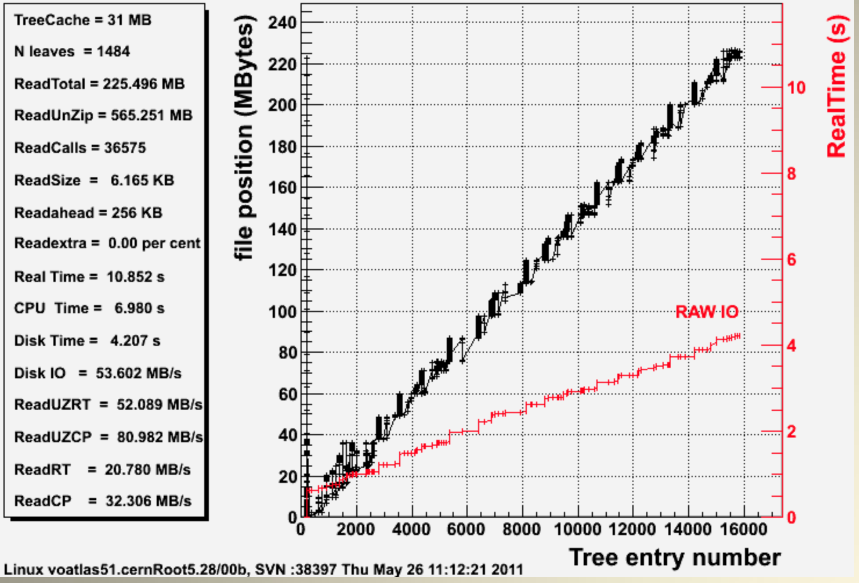
		Real time[s]	CPU time[s]	HDD reads	Transferred [MB]	HDD time [s]
Egamma	100	93.32	87.67	5060	254	11
	10	22.46	17.06	5428	254	11
	1	14.31	8.74	4987	237	10
JetMEt	100	193.61	163.34	31354	1378	63
	10	100.31	62.37	37579	1370	72
	1	65.15	34.52	26309	922	57
Susy	100	19.86	17.08	3149	157	5
	10	11.32	7.74	3532	157	6
	1	10.05	6.25	3687	156	6
Photon	100	17.52	14.55	3613	216	6
	10	11.37	7.81	3705	216	7
	1	10.34	6.93	3770	216	7





D3PD

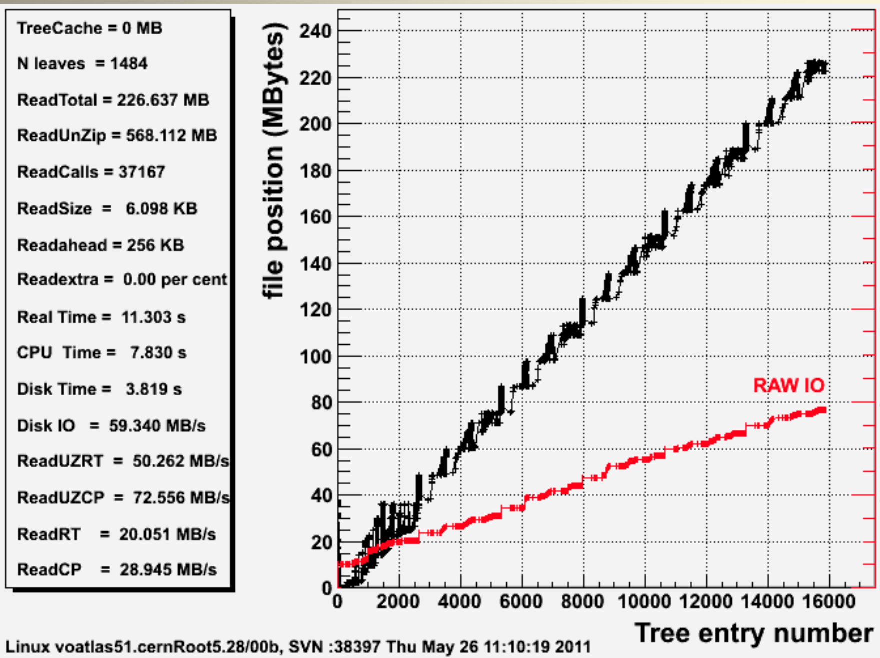
1% events



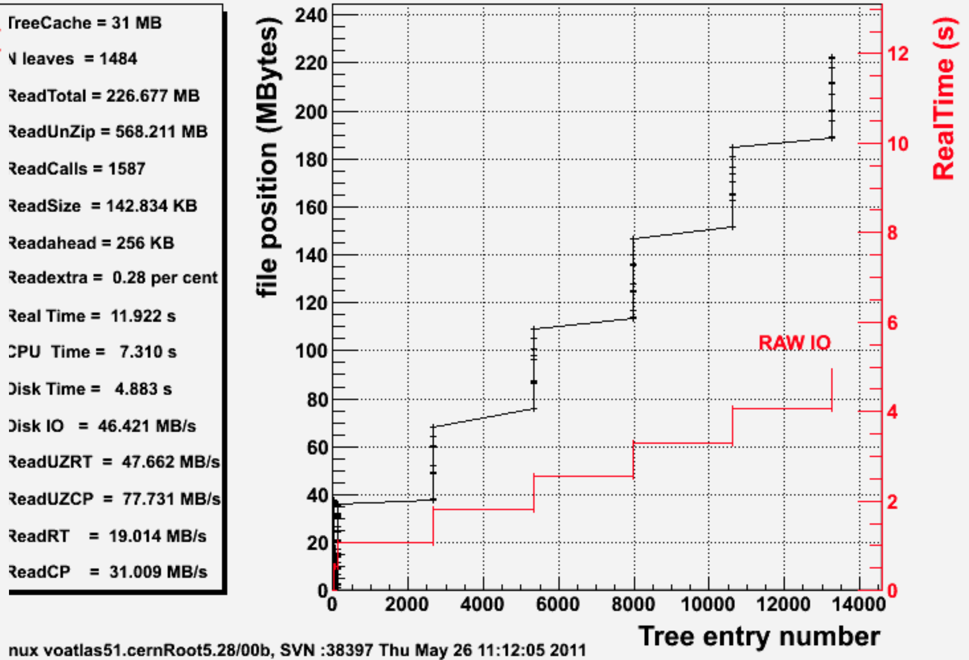
```
TreeCache = 31 MB
N leaves  = 1484
ReadTotal = 225.496 MB
ReadUnZip = 565.251 MB
ReadCalls = 36575
ReadSize  =   6.165 KB
Readahead = 256 KB
Readextra =  0.00 per cent
Real Time =  10.852 s
CPU  Time =   6.980 s
Disk Time =   4.207 s
Disk IO   =  53.602 MB/s
ReadUZRT  =  52.089 MB/s
ReadUZCP  =  80.982 MB/s
ReadRT    =  20.780 MB/s
ReadCP    =  32.306 MB/s
```

In this case TTC
doesn’t learn
as first 100 events not read!

10% events



```
TreeCache = 0 MB
N leaves  = 1484
ReadTotal = 226.637 MB
ReadUnZip = 568.112 MB
ReadCalls = 37167
ReadSize  =   6.098 KB
Readahead = 256 KB
Readextra =  0.00 per cent
Real Time =  11.303 s
CPU  Time =   7.830 s
Disk Time =   3.819 s
Disk IO   =  59.340 MB/s
ReadUZRT  =  50.262 MB/s
ReadUZCP  =  72.556 MB/s
ReadRT    =  20.051 MB/s
ReadCP    =  28.945 MB/s
```



```
TreeCache = 31 MB
N leaves  = 1484
ReadTotal = 226.677 MB
ReadUnZip = 568.211 MB
ReadCalls = 1587
ReadSize  = 142.834 KB
Readahead = 256 KB
Readextra =  0.28 per cent
Real Time =  11.922 s
CPU  Time =   7.310 s
Disk Time =   4.883 s
Disk IO   =  46.421 MB/s
ReadUZRT  =  47.662 MB/s
ReadUZCP  =  77.731 MB/s
ReadRT    =  19.014 MB/s
ReadCP    =  31.009 MB/s
```

Near Future

- Basket optimization needs to be fixed (or go back to basket reordering)
- Need to retest all the read/write scenarios for 17.0.1
- Re-establishing automated IO tests
  - Local disk /LAN
  - WAN? – Not really tried at all – unlike CMS.
  - HammerCloud – Regular test job sent to all sites
  - Measuring efficiencies for real jobs.

Further future

- What can be done to optimise collections:
  - Further improve P-T converters ?
  - Reduce complexity of objects e.g. “AllCells” collection that is a vector of ints.
- What can be done to improve ROOT speeds?
- Many potential improvements in user analysis / D3PD reading codes.