



ROOT I/O
recent improvements
Bottlenecks and Solutions

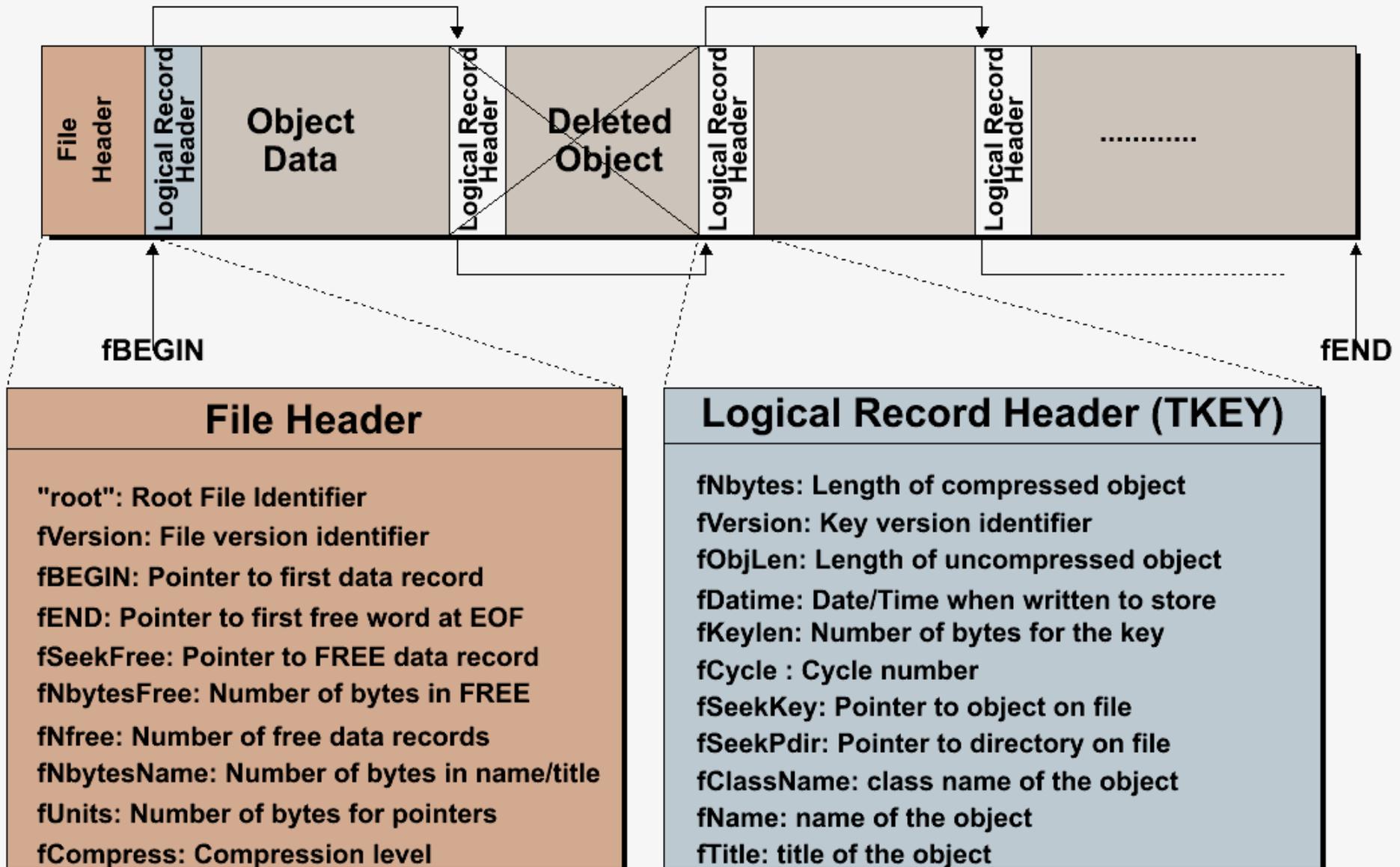
Rene Brun
December 02 2009

Main Points

-  **ROOT IO basics**
-  **Bottlenecks analysis**
-  **Solutions**

ROOT IO Basics

ROOT File description



TFile::Map

```
root [0] TFile *falice = TFile::Open("http://root.cern.ch/files/alice_ESDs.root")
root [1] falice->Map()
20070713/195136  At:100      N=120      TFile
20070713/195531  At:220      N=274      TH1D          CX = 7.38
20070713/195531  At:494      N=331      TH1D          CX = 2.46
20070713/195531  At:825      N=290      TH1D          CX = 2.80
...
20070713/195531  At:391047   N=1010     TH1D          CX = 3.75
Address = 392057      Nbytes = -889  =====G A P=====
20070713/195519  At:392946   N=2515     TBasket       CX = 195.48
20070713/195519  At:395461   N=23141    TBasket       CX = 1.31
20070713/195519  At:418602   N=2566     TBasket       CX = 10.40
20070713/195520  At:421168   N=2518     TBasket       CX = 195.24
20070713/195532  At:423686   N=2515     TBasket       CX = 195.48
20070713/195532  At:426201   N=2023     TBasket       CX = 15.36
20070713/195532  At:428224   N=2518     TBasket       CX = 195.24
20070713/195532  At:430742   N=375281   TTree         CX = 4.28
20070713/195532  At:806023   N=43823    TTree         CX = 100.84
20070713/195532  At:849846   N=6340     TH2F          CX = 100.63
20070713/195532  At:856186   N=951      TH1F          CX = 9.02
20070713/195532  At:857137   N=16537    StreamerInfo  CX = 3.74
20070713/195532  At:873674   N=1367     KeysList
20070713/195532  At:875041   N=1        END
root [2]
```

Classes
dictionary

List of keys

TFile::ls

```
root [3] falice->ls()
KEY: TH1D      logTRD_backfit;1
KEY: TH1D      logTRD_refit;1
KEY: TH1D      logTRD_clSearch;1
KEY: TH1D      logTRD_X;1
KEY: TH1D      logTRD_ncl;1
KEY: TH1D      logTRD_nclTrack;1
KEY: TH1D      logTRD_minYPos;1
KEY: TH1D      logTRD_minYNeg;1
KEY: TH2D      logTRD_minD;1
KEY: TH1D      logTRD_minZ;1
KEY: TH1D      logTRD_deltaX;1
KEY: TH1D      logTRD_xCl;1
KEY: TH1D      logTRD_clY;1
KEY: TH1D      logTRD_clZ;1
KEY: TH1D      logTRD_clB;1
KEY: TH1D      logTRD_clG;1
KEY: TTree     esdTree;1      Tree with ESD objects
KEY: TTree     HLTesdTree;1  Tree with HLT ESD objects
KEY: TH2F      TOFDig_ClusMap;1
KEY: TH1F      TOFDig_NClus;1
KEY: TH1F      TOFDig_ClusTime;1
KEY: TH1F      TOFDig_ClusToT;1
KEY: TH1F      TOFRec_NClusW;1
KEY: TH1F      TOFRec_Dist;1
KEY: TH2F      TOFDig_SigYVsP;1
KEY: TH2F      TOFDig_SigZVsP;1
KEY: TH2F      TOFDig_SigYVsPWin;1
KEY: TH2F      TOFDig_SigZVsPWin;1
```

Shows the list of objects
In the current directory
(like in a file system)

Self-describing files

- 🌐 Dictionary for persistent classes written to the file.
- 🌐 ROOT files can be read by foreign readers
- 🌐 Support for **Backward** and **Forward** compatibility
- 🌐 Files created in 2001 must be readable in 2015
- 🌐 Classes (data objects) for all objects in a file can be regenerated via **TFile::MakeProject**

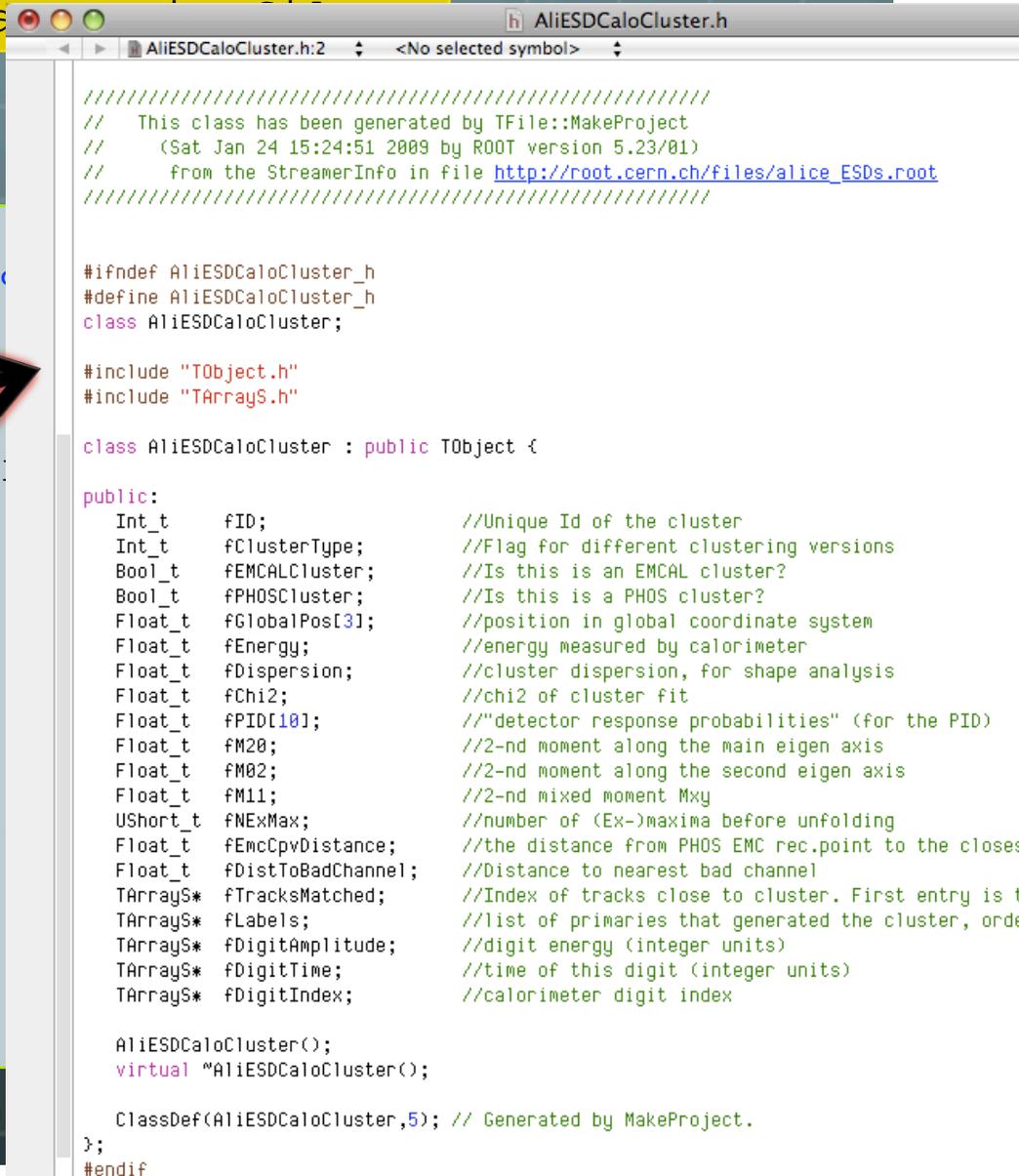
```
Root > TFile f("demo.root");
```

```
Root > f.MakeProject("dir", "*", "new++");
```

TFile::MakeProject

Generate C++ header files
and shared lib for the cla

```
(macbrun2) [253] root
root [0] TFile *falice - TFile::Open("http://ro
root [1] falice->MakeProject("alice","*","++");
MakeProject has generated 26 classes in alice
alice/MAKEP file has been generated
Shared lib alice/alice.so has been generated
Shared lib alice/alice.so has been dynamically
root [2] !ls alice
AliESDCaloCluster.h      AliESDZDC.h
AliESDCaloTrigger.h     AliESDcascade.h
AliESDEvent.h           AliESDfriend.h
AliESDFMD.h             AliESDfriendTrack.h
AliESDHeader.h          AliESDkink.h
AliESDMuonTrack.h       AliESDtrack.h
AliESDPmdTrack.h        AliESDv0.h
AliESDRun.h             AliExternalTrackParam.h
AliESDTZERO.h           AliFMDFloatMap.h
AliESDTrdTrack.h        AliFMDFMap.h
AliESDVZERO.h           AliMultiplicity.h
AliESDVertex.h          AliRawDataErrorLog.h
root [3]
```



```
AliESDCaloCluster.h
AliESDCaloCluster.h:2  <No selected symbol>

////////////////////////////////////
// This class has been generated by TFile::MakeProject
// (Sat Jan 24 15:24:51 2009 by ROOT version 5.23/01)
// from the StreamerInfo in file http://root.cern.ch/files/alice\_ESDs.root
////////////////////////////////////

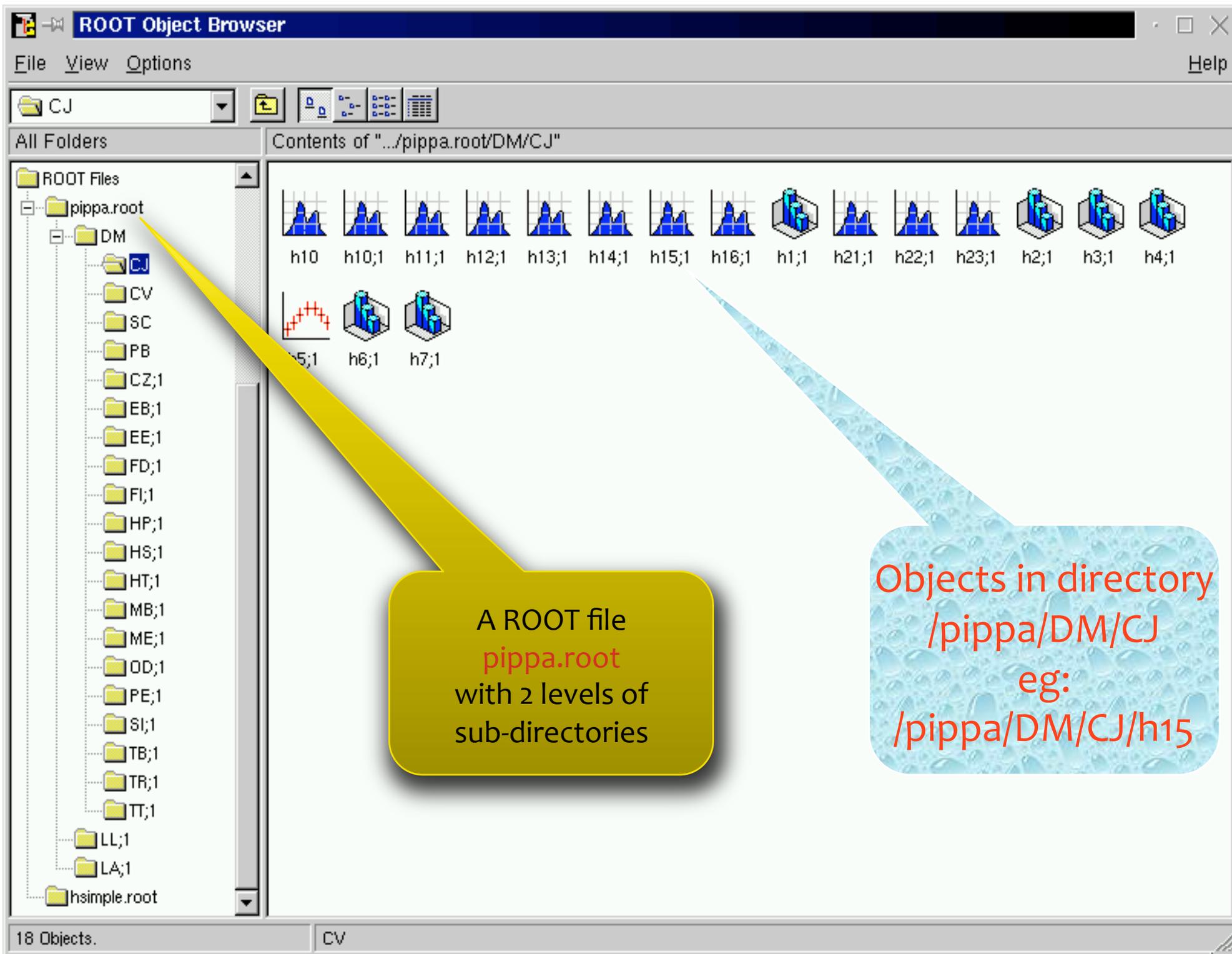
#ifndef AliESDCaloCluster_h
#define AliESDCaloCluster_h
class AliESDCaloCluster;

#include "TObject.h"
#include "TArrayS.h"

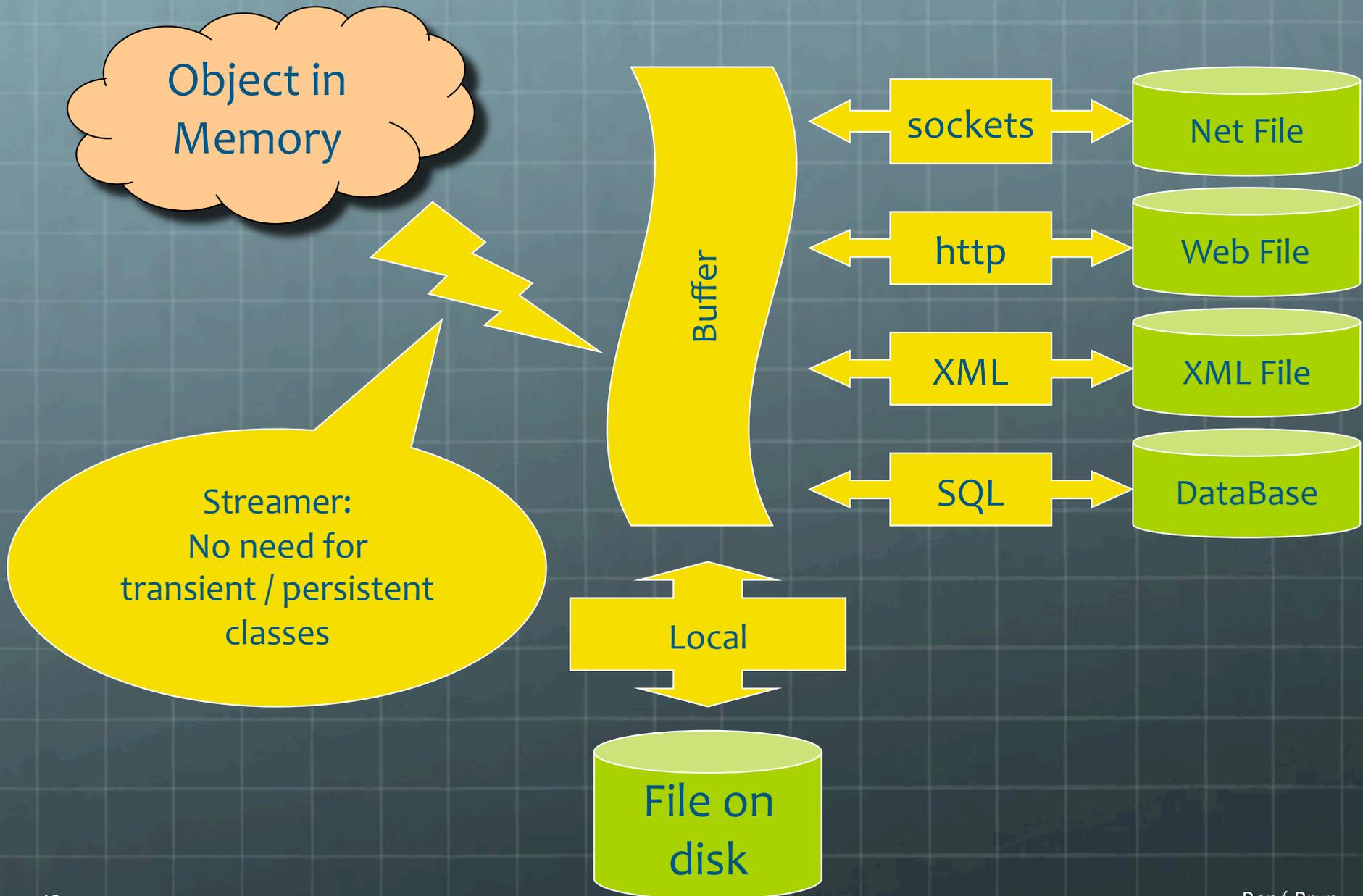
class AliESDCaloCluster : public TObject {
public:
    Int_t      fID; //Unique Id of the cluster
    Int_t      fClusterType; //Flag for different clustering versions
    Bool_t     fEMCALcluster; //Is this is an EMCAL cluster?
    Bool_t     fPHOScluster; //Is this is a PHOS cluster?
    Float_t    fGlobalPos[3]; //position in global coordinate system
    Float_t    fEnergy; //energy measured by calorimeter
    Float_t    fDispersion; //cluster dispersion, for shape analysis
    Float_t    fChi2; //chi2 of cluster fit
    Float_t    fPID[10]; //detector response probabilities" (for the PID)
    Float_t    fM20; //2-nd moment along the main eigen axis
    Float_t    fM02; //2-nd moment along the second eigen axis
    Float_t    fM11; //2-nd mixed moment Mxy
    UShort_t   fNExMax; //number of (Ex-)maxima before unfolding
    Float_t    fEmcCpvDistance; //the distance from PHOS EMC rec.point to the closest
    Float_t    fDistToBadChannel; //Distance to nearest bad channel
    TArrayS*   fTracksMatched; //Index of tracks close to cluster. First entry is t
    TArrayS*   fLabels; //list of primaries that generated the cluster, orde
    TArrayS*   fDigitAmplitude; //digit energy (integer units)
    TArrayS*   fDigitTime; //time of this digit (integer units)
    TArrayS*   fDigitIndex; //calorimeter digit index

    AliESDCaloCluster();
    virtual ~AliESDCaloCluster();

    ClassDef(AliESDCaloCluster,5); // Generated by MakeProject.
};
#endif
```

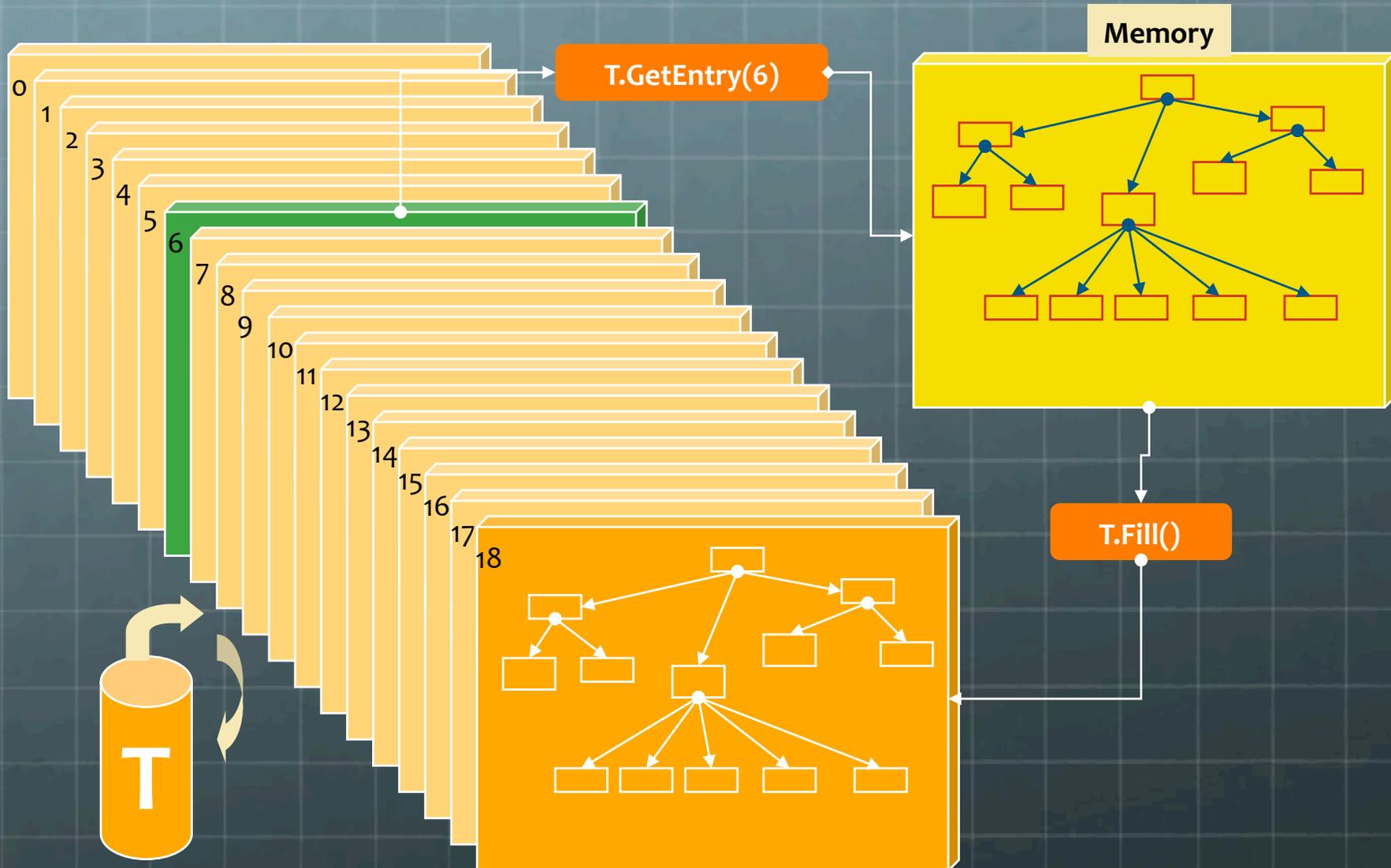


I/O



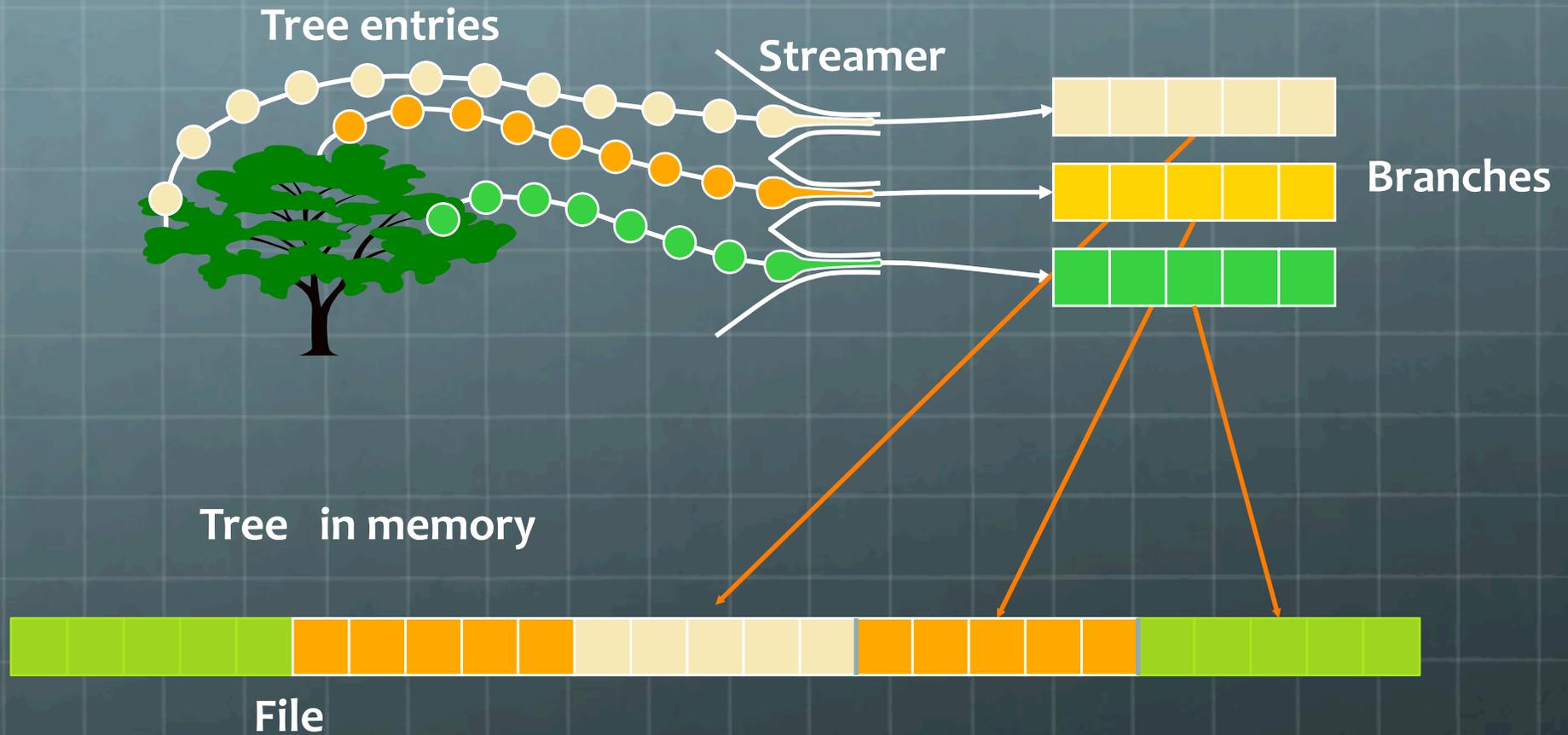
Memory <--> Tree

Each Node is a branch in the Tree



ROOT I/O -- *Split/Cluster*

Tree version



Browsing a TTree with TBrowser

The screenshot displays the ROOT Object Browser interface. The window title is "ROOT Object Browser". The menu bar includes "File", "View", "Options", and "Help". The current directory is "Electrons". The left pane, "All Folders", shows a tree structure with "Electrons" selected under "T". The right pane, "Contents of \"../atlfast.root/T/Electrons\"", lists eight files: "Electrons.fBits", "Electrons.fUniqueID", "Electrons.m_Eta", "Electrons.m_KFcode", "Electrons.m_KFmother", "Electrons.m_MCParticle", "Electrons.m_PT", and "Electrons.m_Phi".

Callouts provide additional information:

- A blue callout bubble with a water droplet pattern contains the text "8 leaves of branch Electrons".
- A yellow callout bubble points to the "Electrons" folder in the left pane with the text "8 branches of T".
- A yellow callout bubble points to the "Electrons.m_PT" file in the right pane with the text "A double click To histogram The leaf".

At the bottom left of the window, it says "8 Objects."

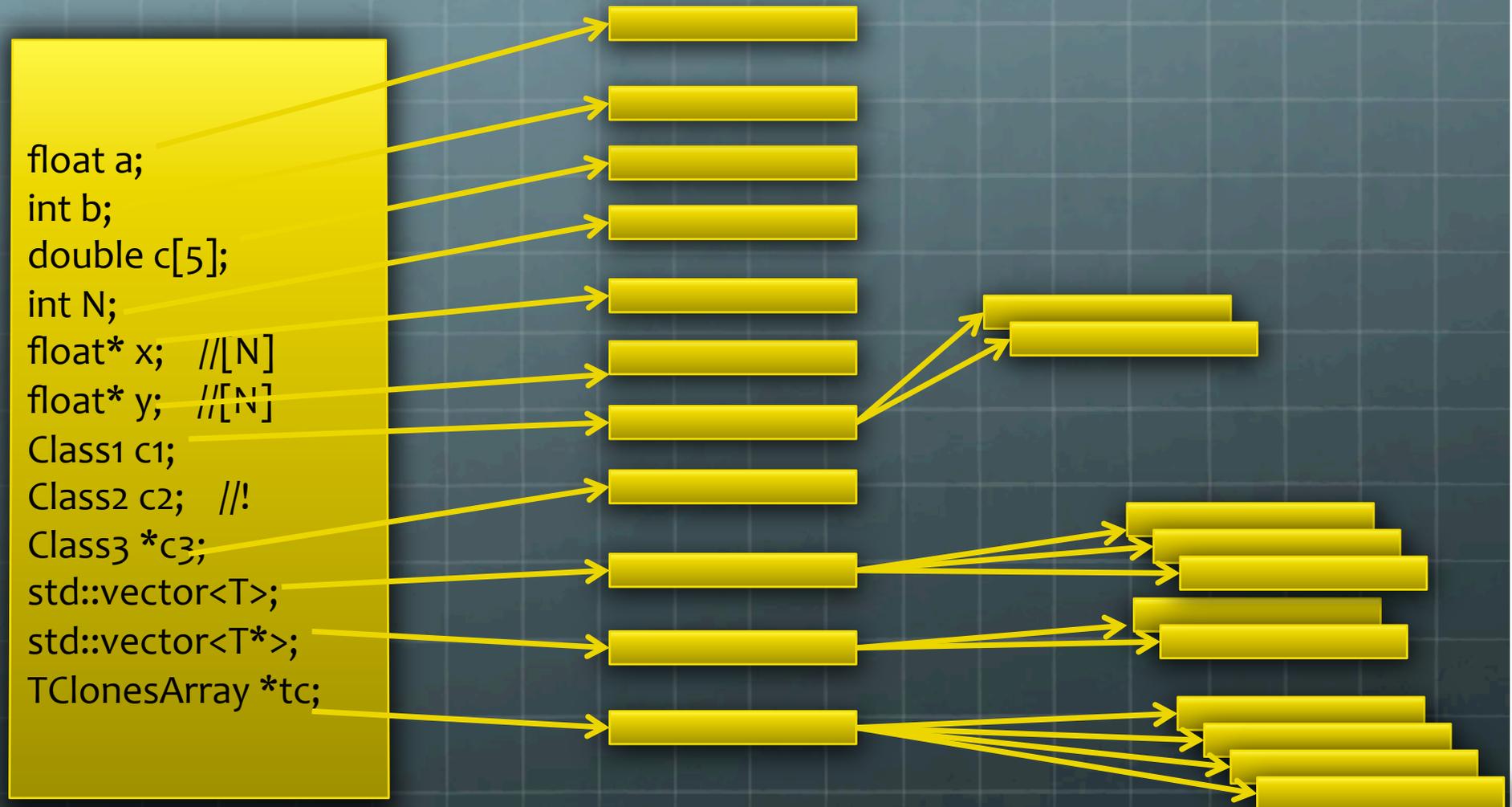
Trees Split Mode

**Object-wise
Member-wise
streaming**

Creating branches

-  **B1:** Branches can be created automatically from the natural hierarchy in the top level object.
-  **B2:** Branches can be created automatically from a dynamic list of top level objects (recommended).
-  **B3:** Branches can be created manually one by one specifying a top level object per top branch.
-  **B4:** Sub-branches are automatically created for **STL collections** and **TClonesArray**.

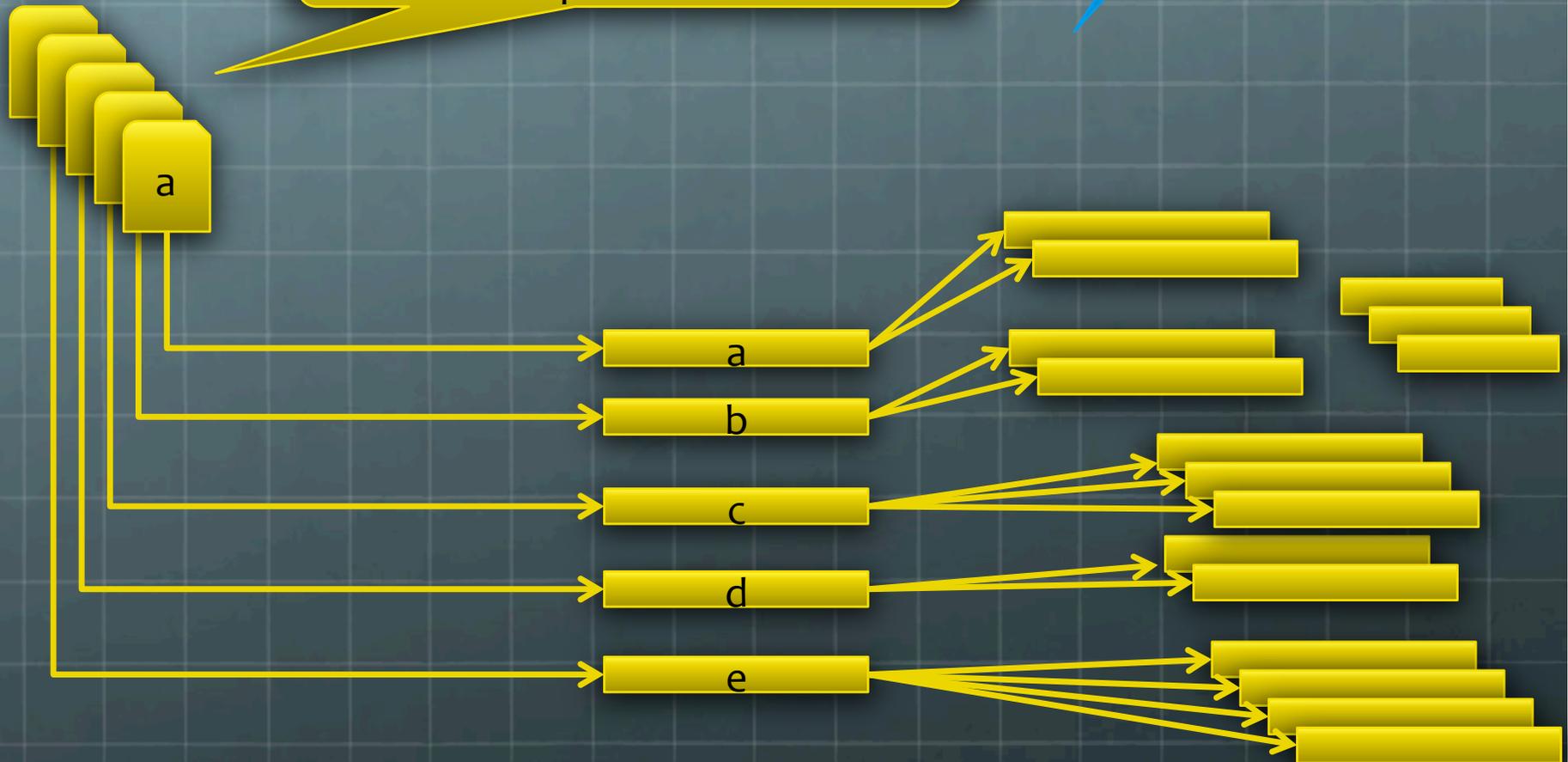
Case B1



Case B2

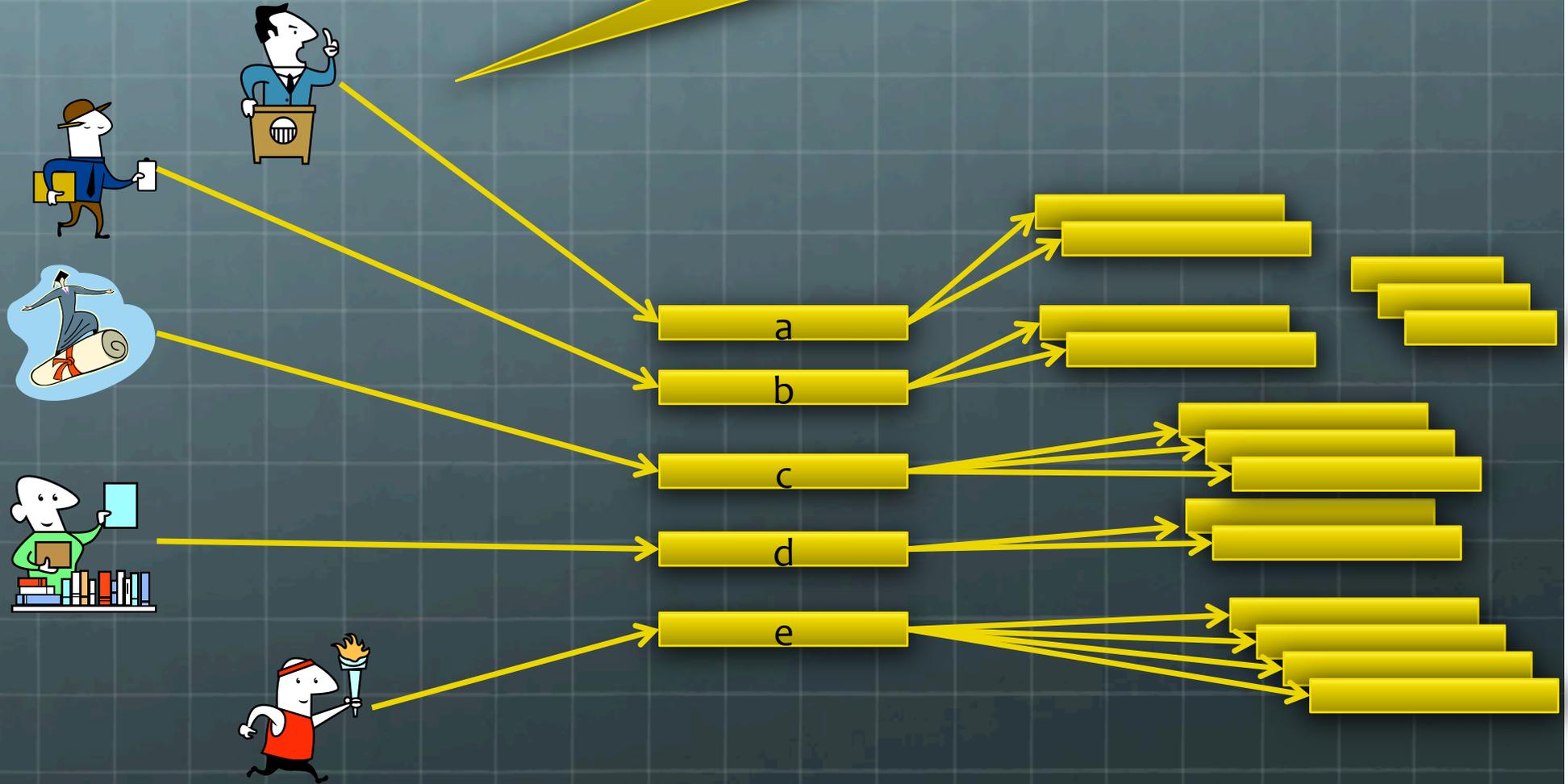
Best solution

Each named member of a TList
Is a top level branch



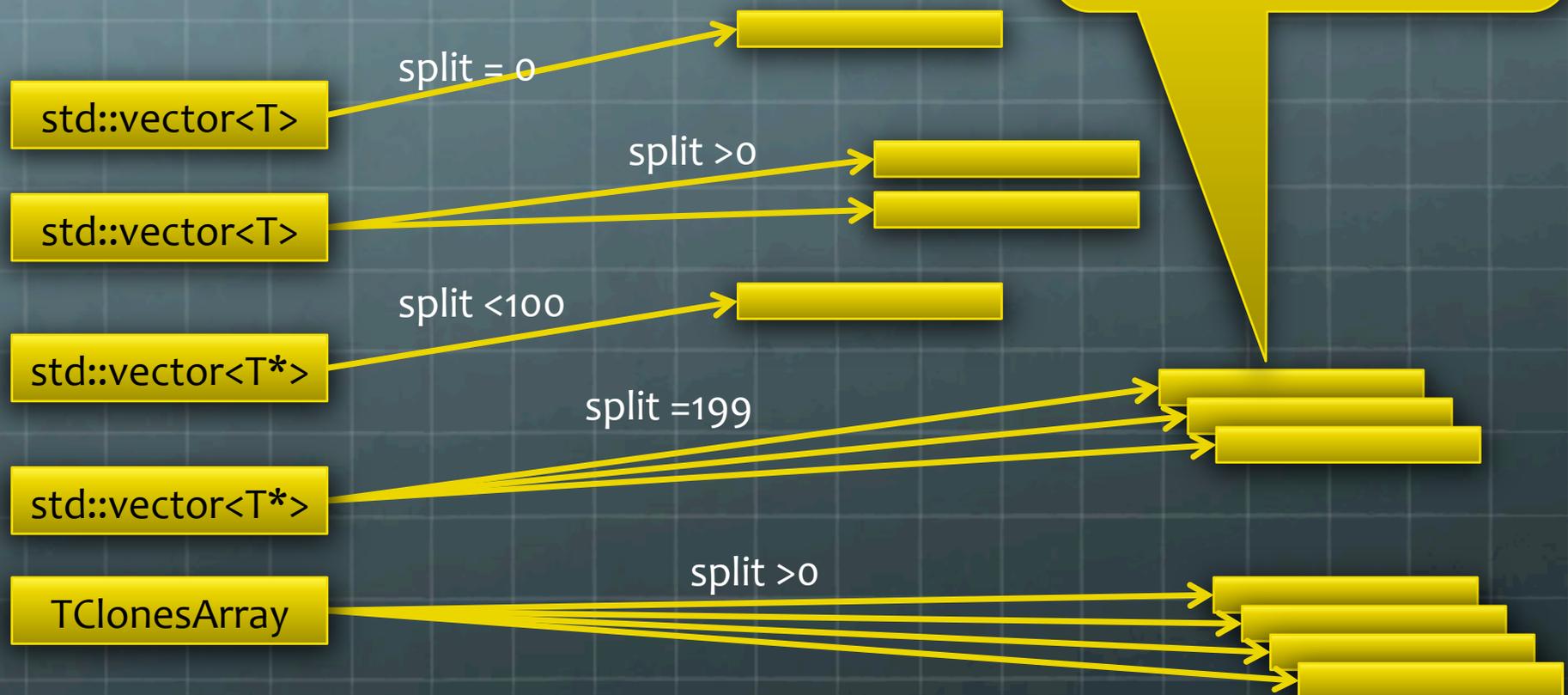
Case B3

Top level branches
Are created manually



Case B4 Collections

Use split=199 to reduce
The number of branches
In case of polymorphic
collections



Collections are not equal

Time to read in seconds	comp 0	comp 1

TClonesArray(TObjHit) level=99	0.14	0.45
TClonesArray(TObjHit) level= 0	0.17	0.63
vector<THit> level=99	0.32	0.52
vector<THit> level= 0	0.48	0.87
vector<THit> level= 0 MW	0.23	0.65
list<THit> level=99	0.52	0.64
list<THit> level= 0	0.73	0.95
list<THit> level= 0 MW	0.49	0.62
deque<THit> level=99	0.31	0.58
deque<THit> level= 0	0.55	0.93
deque<THit> level= 0 MW	0.36	0.62
set<THit> level=99	0.46	0.81
set<THit> level= 0	0.74	1.06
set<THit> level= 0 MW	0.44	0.69
multiset<THit> level=99	0.46	0.84
multiset<THit> level= 0	0.72	1.06
multiset<THit> level= 0 MW	0.45	0.70
map<THit> level=99	0.45	0.76
map<THit> level= 0	0.97	1.12
map<THit> level= 0 MW	0.84	1.14

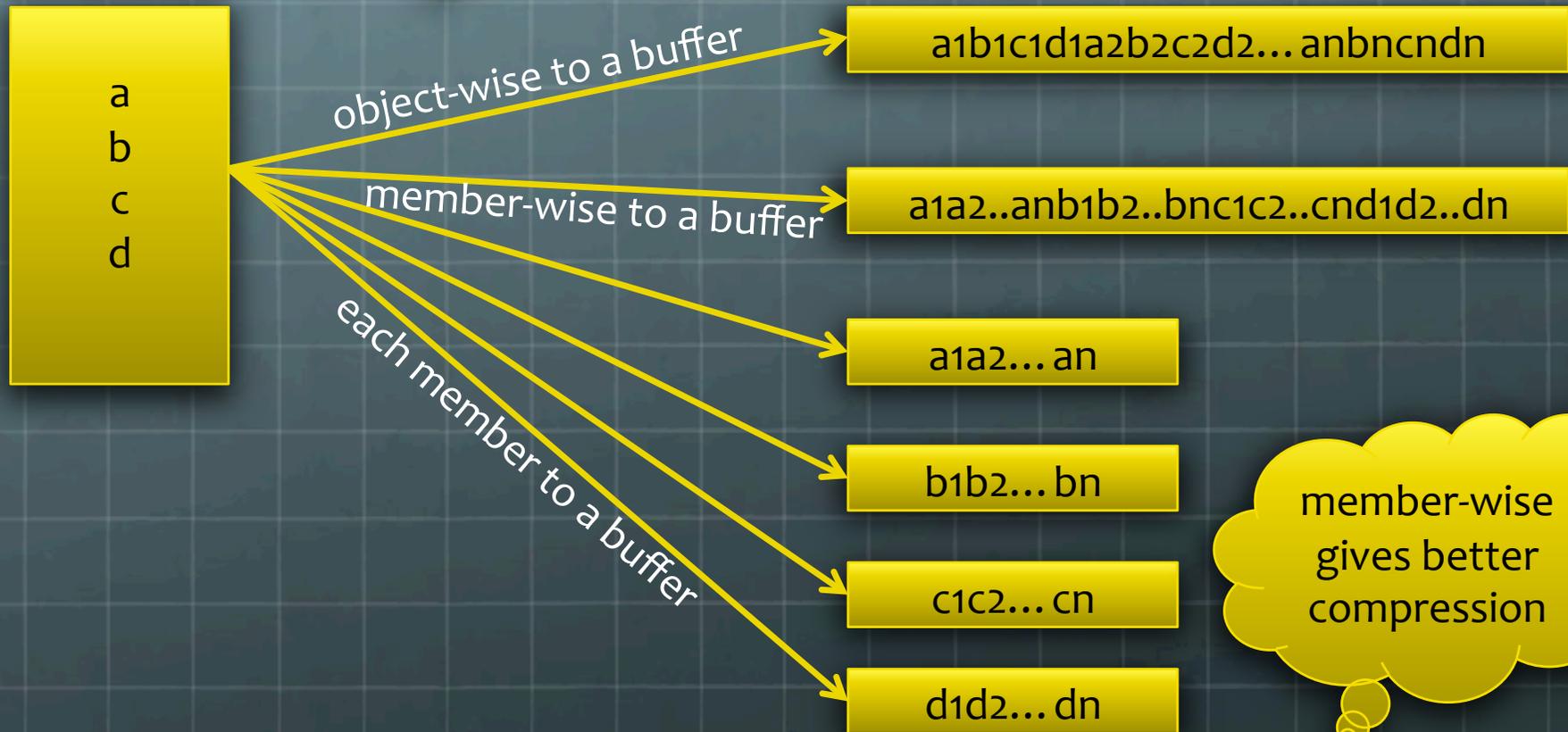
Time to read in seconds	comp 0	comp 1

multimap<THit> level=99	0.45	0.78
multimap<THit> level= 0	0.89	1.11
multimap<THit> level= 0 MW	0.79	1.12
vector<THit*> level=25599	0.34	0.61
list<THit*> level=25599	0.41	0.81
deque<THit*> level=25599	0.49	0.62
set<THit*> level=25599	0.59	0.76
multiset<THit*> level=25599	0.49	0.73
map<THit*> level=99	1.41	1.55
multimap<THit*> level=99	1.15	1.54
vector<THit*> level=99 (NS)	1.17	1.38
list<THit*> level=99 (NS)	1.33	1.48
deque<THit*> level=99 (NS)	1.36	1.39
set<THit*> level=99 (NS)	1.02	1.48
multiset<THit*> level=99 (NS)	1.28	1.45

For more details run `$ROOTSYS/test/bench`

ObjectWise/MemberWise Streaming

3 modes to stream
an object



member-wise
gives better
compression

More branches is better

- 🌐 Better compression
- 🌐 Faster when reading (in particular a subset of branches)
- 🌐 Good for parallelism (for the future)

10000 is may be too much 😊

Split mode	File size (MBytes)	Comp factor	Write time (seconds)	Read time (seconds)
0 1 branch	177.45	2.23	38.7	12.6
1 20 branches	174.7	2.26	37.9	12.0
99 56 branches	144.9	2.53	38.3	10.8

ROOT IO

Bottlenecks and Solutions

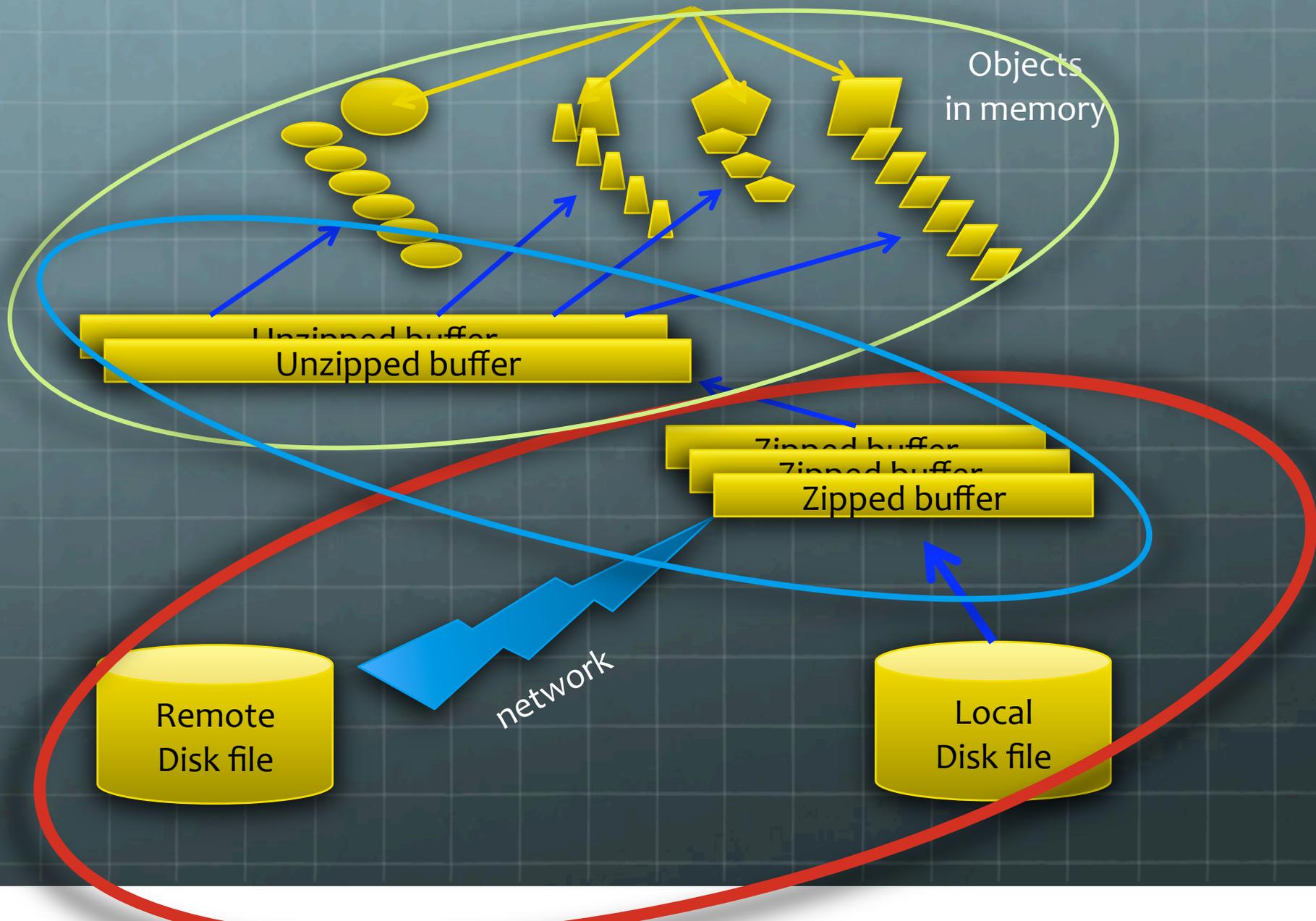
Buffering effects

- 🌐 Branch buffers are not full at the same time.
- 🌐 A branch containing one integer/event and with a buffer size of 32Kbytes will be written to disk every 8000 events, while a branch containing a non-split collection may be written at each event.
- 🌐 This may give serious problems when reading if the file is not read sequentially.

problems

- 🌐 In the past few months we have analyzed Trees from Alice, Atlas, CMS and LHCb and found problems in all cases.
- 🌐 Some of these problems have solutions with old versions of ROOT, eg
 - 🌐 Using effectively the **TreeCache**
 - 🌐 Optimizing basket buffers
- 🌐 ROOT version 5.25/04 contains several enhancements helping to improve size and performance

Important factors



What is the **TreeCache**

- 🌐 It groups into one buffer all blocks from the used branches.
- 🌐 The blocks are sorted in ascending order and consecutive blocks merged such that the file is read sequentially.
- 🌐 It reduces typically by a factor 10000 the number of transactions with the disk and in particular the network with servers like **xrootd** or **dCache**.
- 🌐 The small blocks in the buffer can be unzipped in parallel on a multi-core machine.
- 🌐 The typical size of the **TreeCache** is 10 Mbytes, but higher values will always give better results. If you have no memory problem, set large values like 200 Mbytes.

TreeCache: new interface

- 🌐 Facts: Most users did not know if they were using or not the **TreeCache**.
- 🌐 We decided to implement a simpler interface from **TTree** itself (no need to know about the class **TTreeCache** anymore).
- 🌐 Because some users claimed to use the **TreeCache** and the results clearly showing the contrary, we decided to implement a new IO monitoring class **TTreePerfStats**.

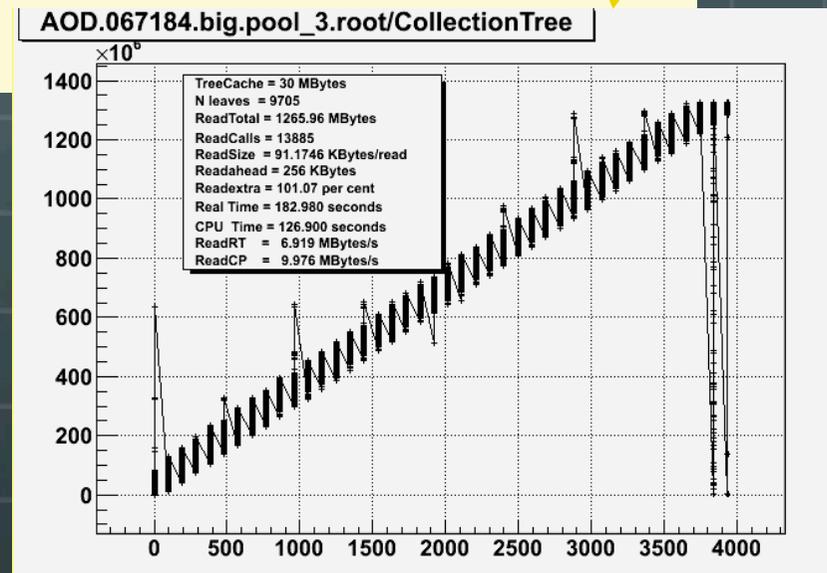
Use TTreePerfStats

```
void taodr(Int_t cachesize=10000000) {
    gSystem->Load("aod/aod"); //shared lib generated with TFile::MakeProject
    TFile *f = TFile::Open("AOD.067184.big.pool.root");
    TTree *T = (TTree*)f->Get("CollectionTree");
    Long64_t nentries = T->GetEntries();
    T->SetCacheSize(cachesize);
    T->AddBranchToCache("*", kTRUE);

    TTreePerfStats ps("ioperf", T);

    for (Long64_t i=0; i<nentries; i++) {
        T->GetEntry(i);
    }
    ps.SaveAs("aodperf.root");
}
```

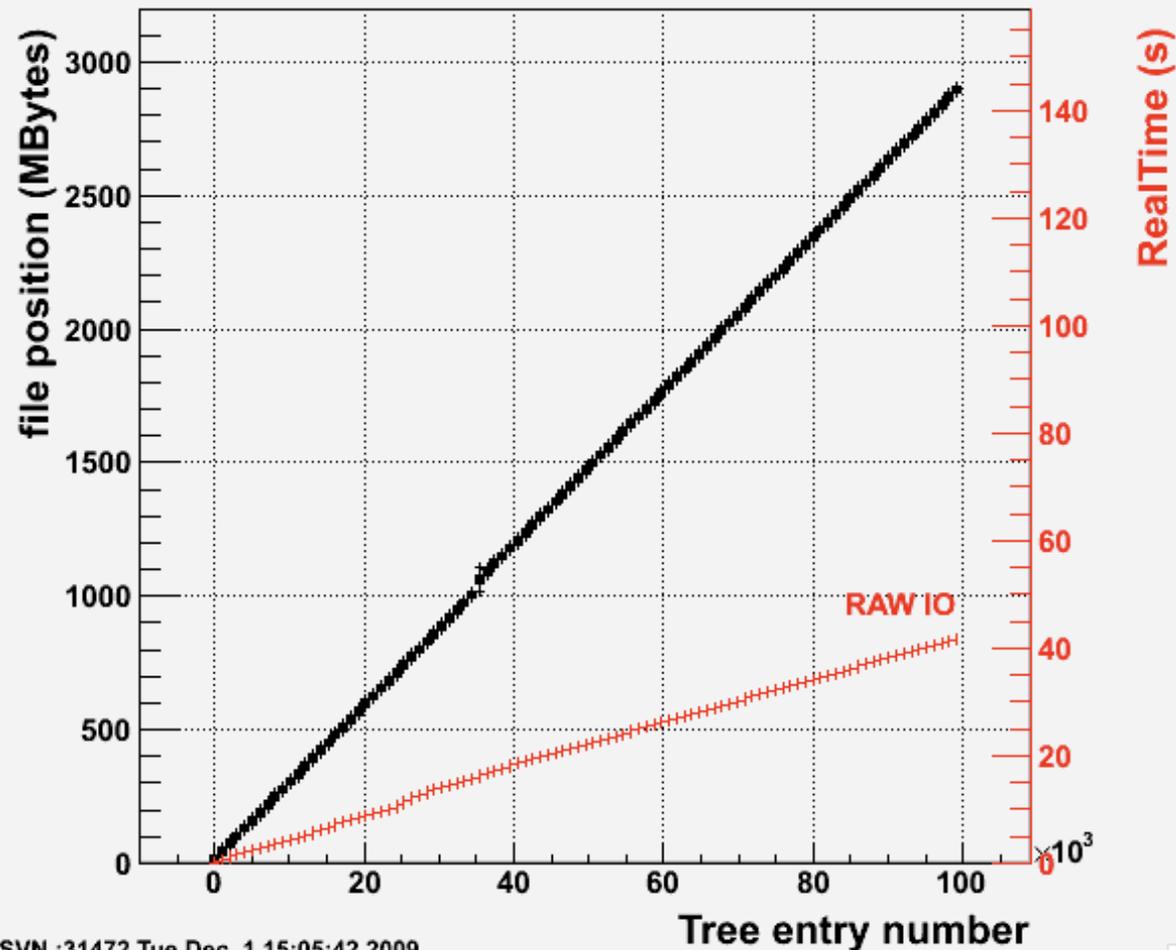
```
Root > TFile f("aodperf.root")
Root > ioperf.Draw()
```



Performance with standard ROOT example

Event100000.root/T

TreeCache = 43 MB
N leaves = 55
ReadTotal = 2874.11 MB
ReadUnZip = 7302.38 MB
ReadCalls = 1260
ReadSize = 2281.040 KB
Readahead = 256 KB
Readextra = 0.01 per cent
Real Time = 144.350 s
CPU Time = 102.570 s
Disk Time = 41.902 s
Disk IO = 68.591 MB/s
ReadUZRT = 50.588 MB/s
ReadUZCP = 71.194 MB/s
ReadRT = 19.911 MB/s
ReadCP = 28.021 MB/s



Test conditions

- 🌐 Because both the **TreeCache** and **Readahead** are designed to minimize the difference RealTime-CpuTime, care has been taken to run the tests with “cold” files, making sure that system buffers were dropped before running a new test.
- 🌐 Note that increasing the **TreeCache** size reduces also the CpuTime.
- 🌐 Note that running **OptimizeBaskets** also reduces substantially the CpuTime because the number of baskets is in general reduced by several factors.

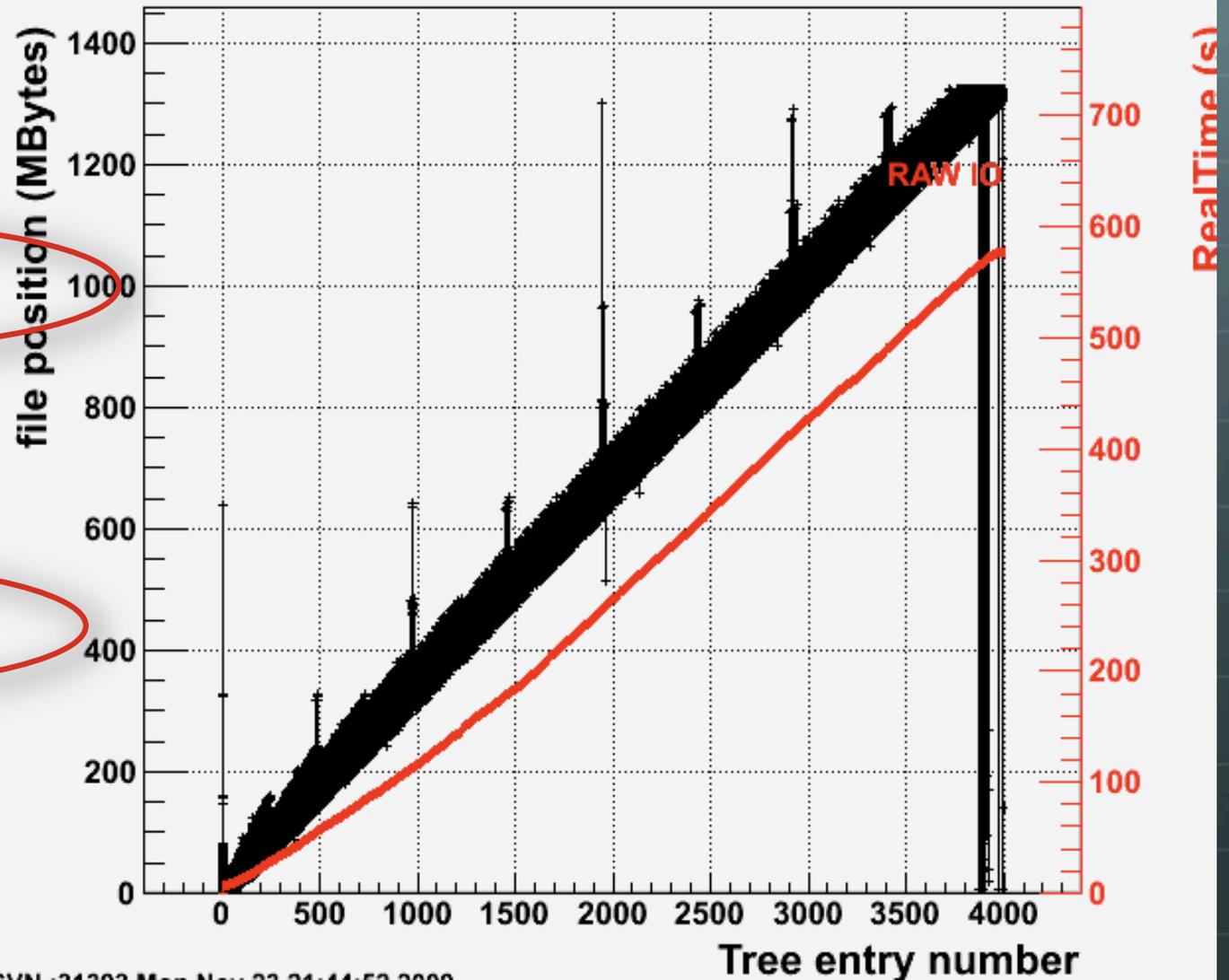
Test conditions 2

- Using one of the AOD files the class headers have been generated automatically via **TTree::MakeProject**.
- The corresponding shared library is linked such that the same object model is used in my tests and in Atlas persistent model.
- The tests read systematically all entries in all branches. Separate tests have been run to check that the optimal performance is still obtained when reading either a subset of branches, a subset of entries or both. This is an important remark because we have seen that sometimes proposed solutions are good when reading everything and very bad in the other mentioned use cases that are typical of the physics analysis scenarios.

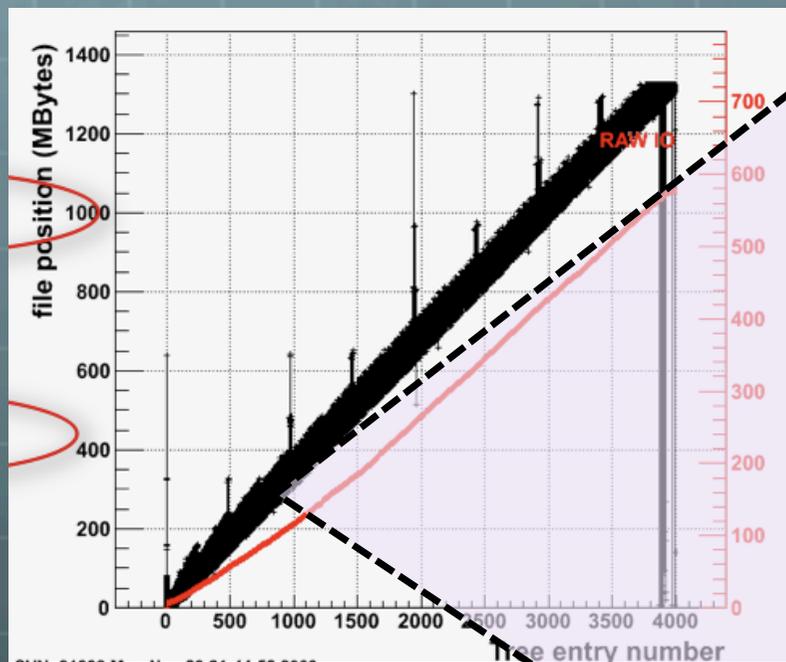
See Doctor

AOD.067184.big.pool_4.root/CollectionTree

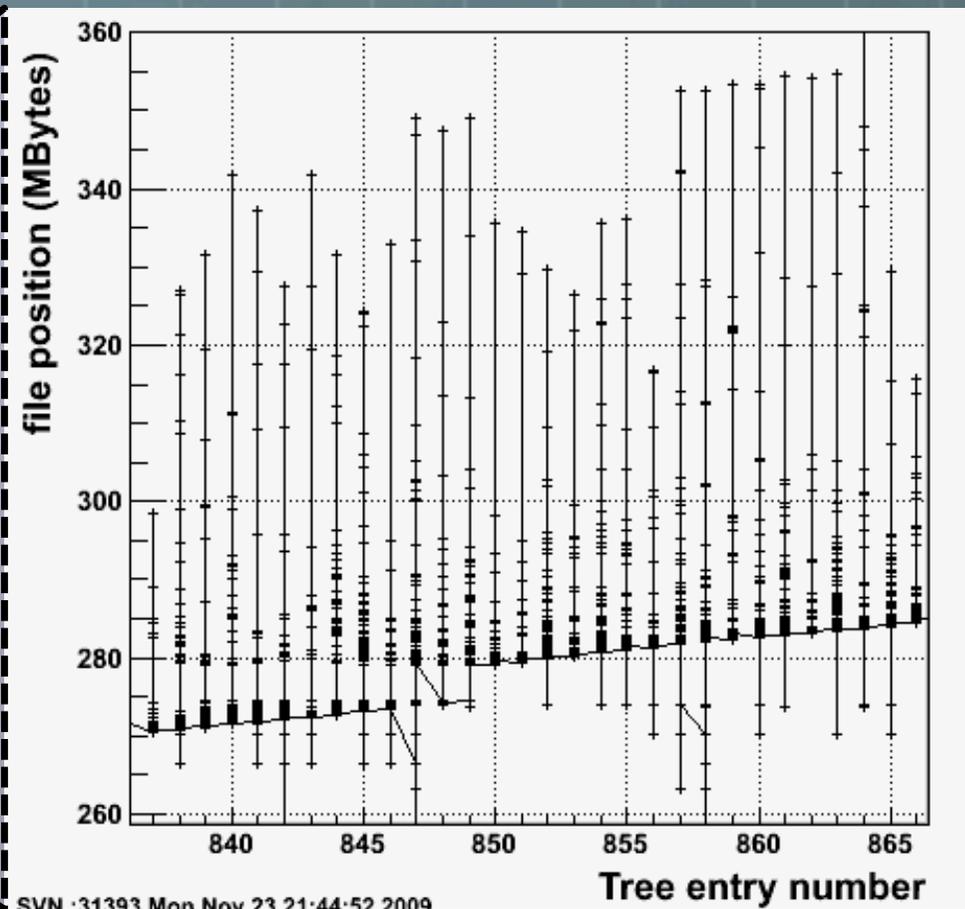
TreeCache = 0 MB
N leaves = 9705
ReadTotal = 1265.92 MB
ReadUnZip = 4057.84 MB
ReadCalls = 1328586
ReadSize = 0.953 KB
Readahead = 256 KB
Readextra = 0.00 per cent
Real Time = 722.315 s
CPU Time = 159.250 s
Disk Time = 577.992 s
Disk IO = 2.190 MB/s
ReadUZRT = 5.618 MB/s
ReadUZCP = 25.481 MB/s
ReadRT = 1.753 MB/s
ReadCP = 7.949 MB/s



Overlapping reads



100 MBytes

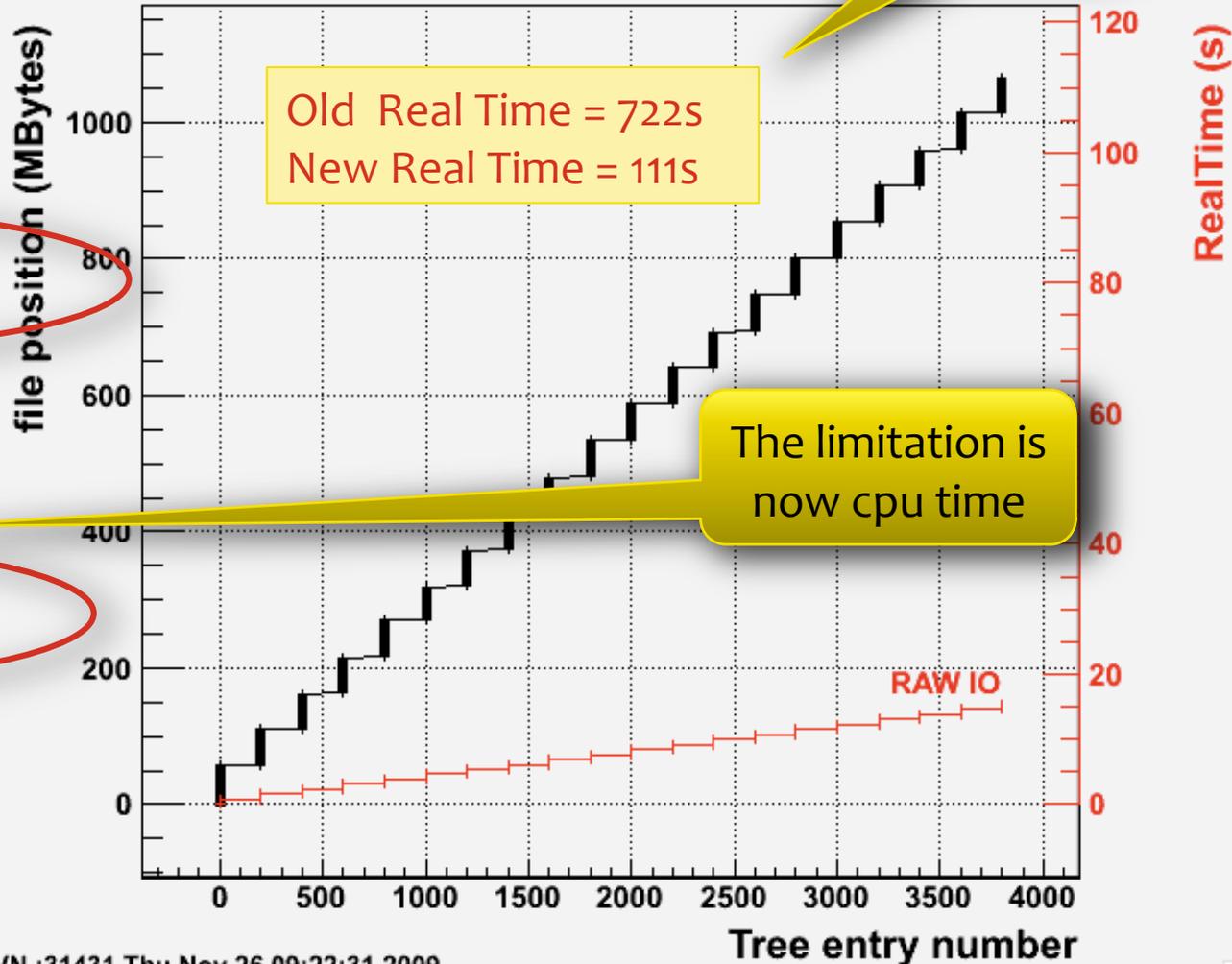


SVN :31393 Mon Nov 23 21:44:52 2009

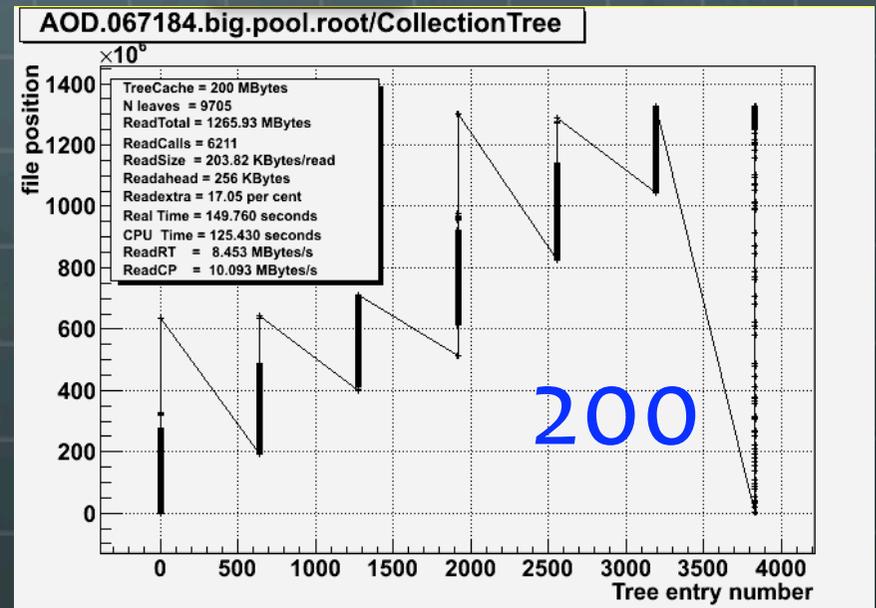
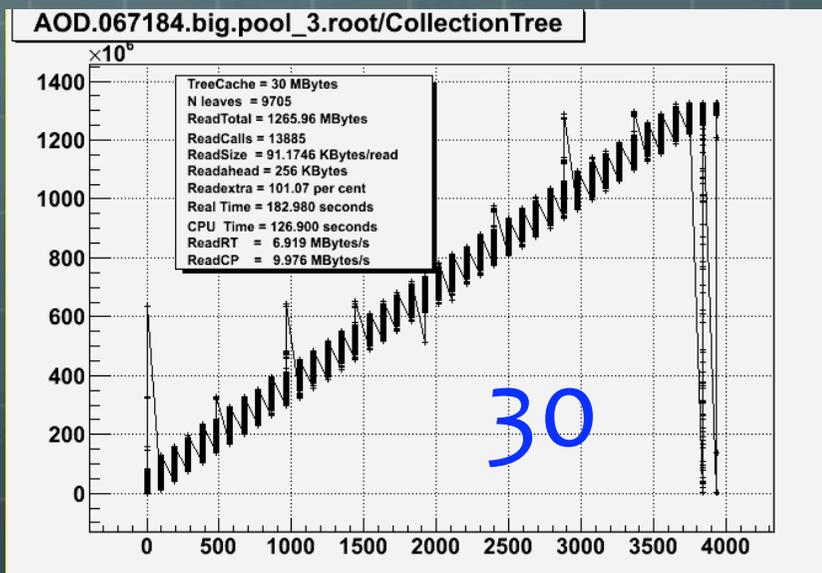
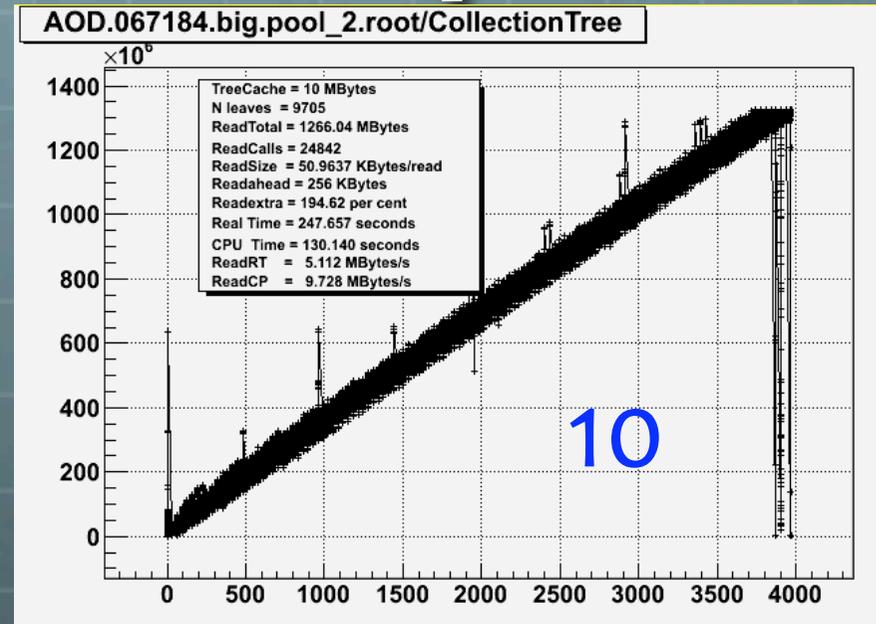
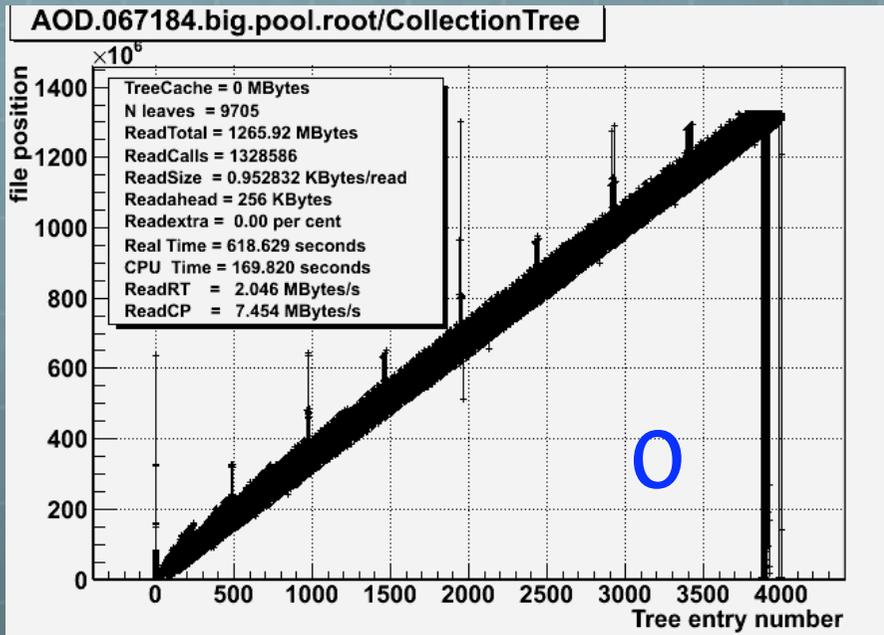
After doctor

atlasFlushed.root/CollectionTree

TreeCache = 60 MB
N leaves = 9705
ReadTotal = 1070.72 MB
ReadUnZip = 3936.2 MB
ReadCalls = 521
ReadSize = 2055.130 KB
Readahead = 256 KB
Readextra = 0.00 per cent
Real Time = 111.563 s
CPU Time = 96.340 s
Disk Time = 15.374 s
Disk IO = 69.645 MB/s
ReadUZRT = 35.282 MB/s
ReadUZCP = 40.857 MB/s
ReadRT = 9.597 MB/s
ReadCP = 11.114 MB/s



TreeCache size impact



TreeCache results table

Original Atlas file (1266MB), 9705 branches split=99

Cache size (MB)	readcalls	RT pcbrun4 (s)	CP pcbrun4 (s)	RT macbrun (s)	CP macbrun (s)
0	1328586	734.6	270.5	618.6	169.8
LAN 1ms 0	1328586	734.6+1300	270.5	618.6+1300	169.8
10	24842	298.5	228.5	229.7	130.1
30	13885	272.1	215.9	183.0	126.9
200	6211	217.2	191.5	149.8	125.4

Reclust: OptimizeBaskets 30 MB (1147 MB), 203 branches split=0

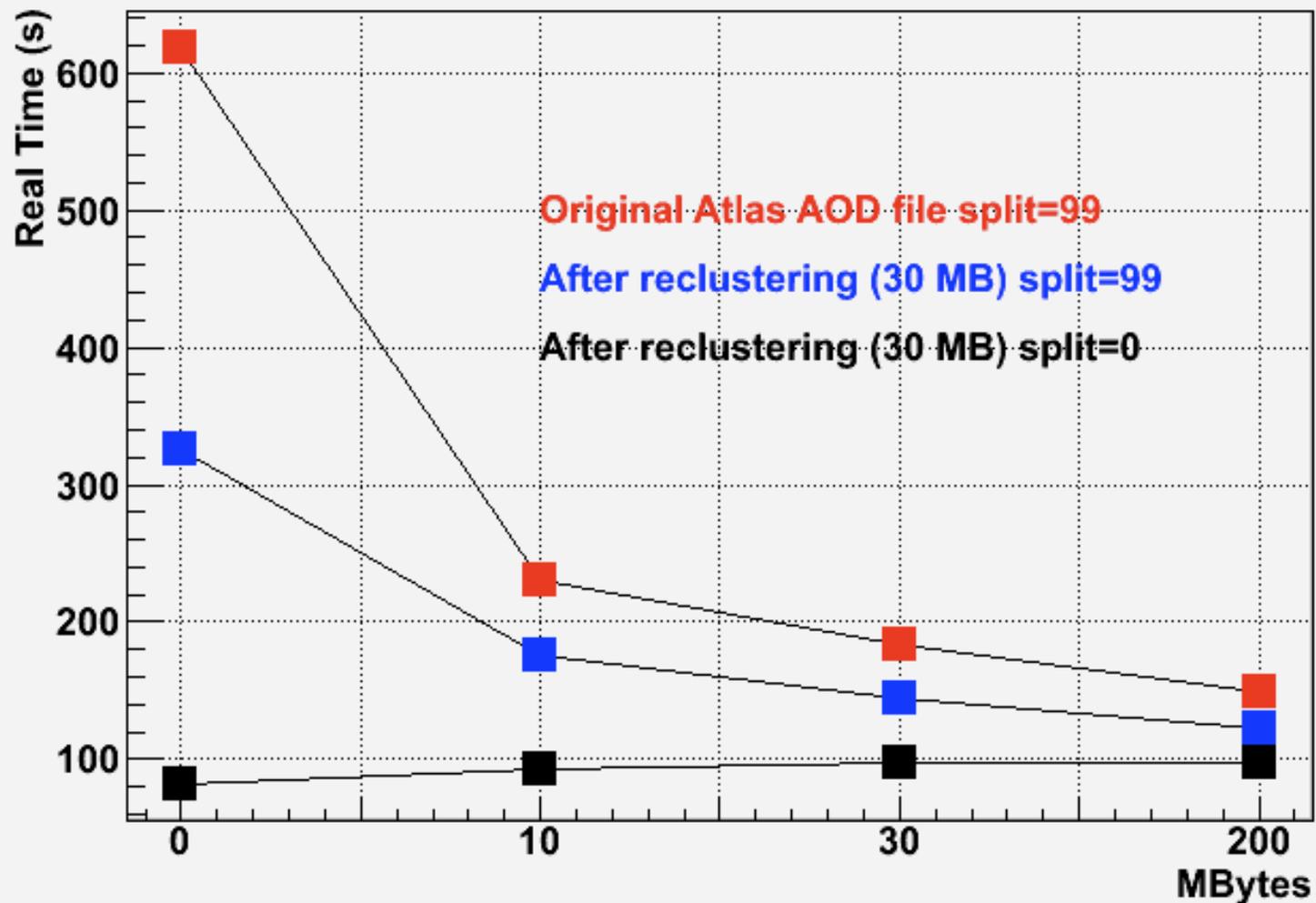
Cache size (MB)	readcalls	RT pcbrun4 (s)	CP pcbrun4 (s)	RT macbrun (s)	CP macbrun (s)
0	15869	148.1	141.4	81.6	80.7
LAN 1ms 0	15869	148.1 + 16	141.4	81.6 + 16	80.7
10	714	157.9	142.4	93.4	82.5
30	600	165.7	148.8	97.0	82.5
200	552	154.0	137.6	98.1	82.0

Reclust: OptimizeBaskets 30 MB (1086 MB), 9705 branches split=99

Cache size (MB)	readcalls	RT pcbrun4 (s)	CP pcbrun4 (s)	RT macbrun (s)	CP macbrun (s)
0	515350	381.8	216.3	326.2	127.0
LAN 1ms 0	515350	381.8 + 515	216.3	326.2 + 515	127.0
10	15595	234.0	185.6	175.0	106.2
30	8717	216.5	182.6	144.4	104.5
200	2096	182.5	163.3	122.3	103.4

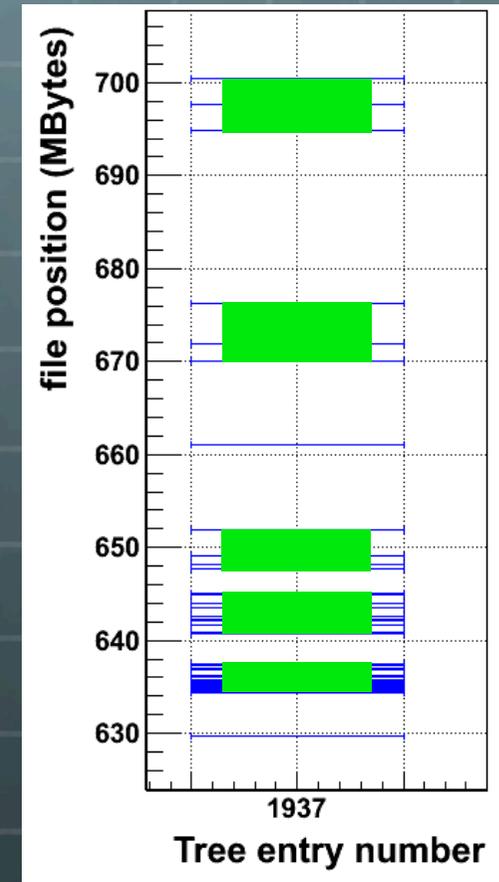
TreeCache results graph

IO speed vs TreeCache size



What is the **readahead cache**

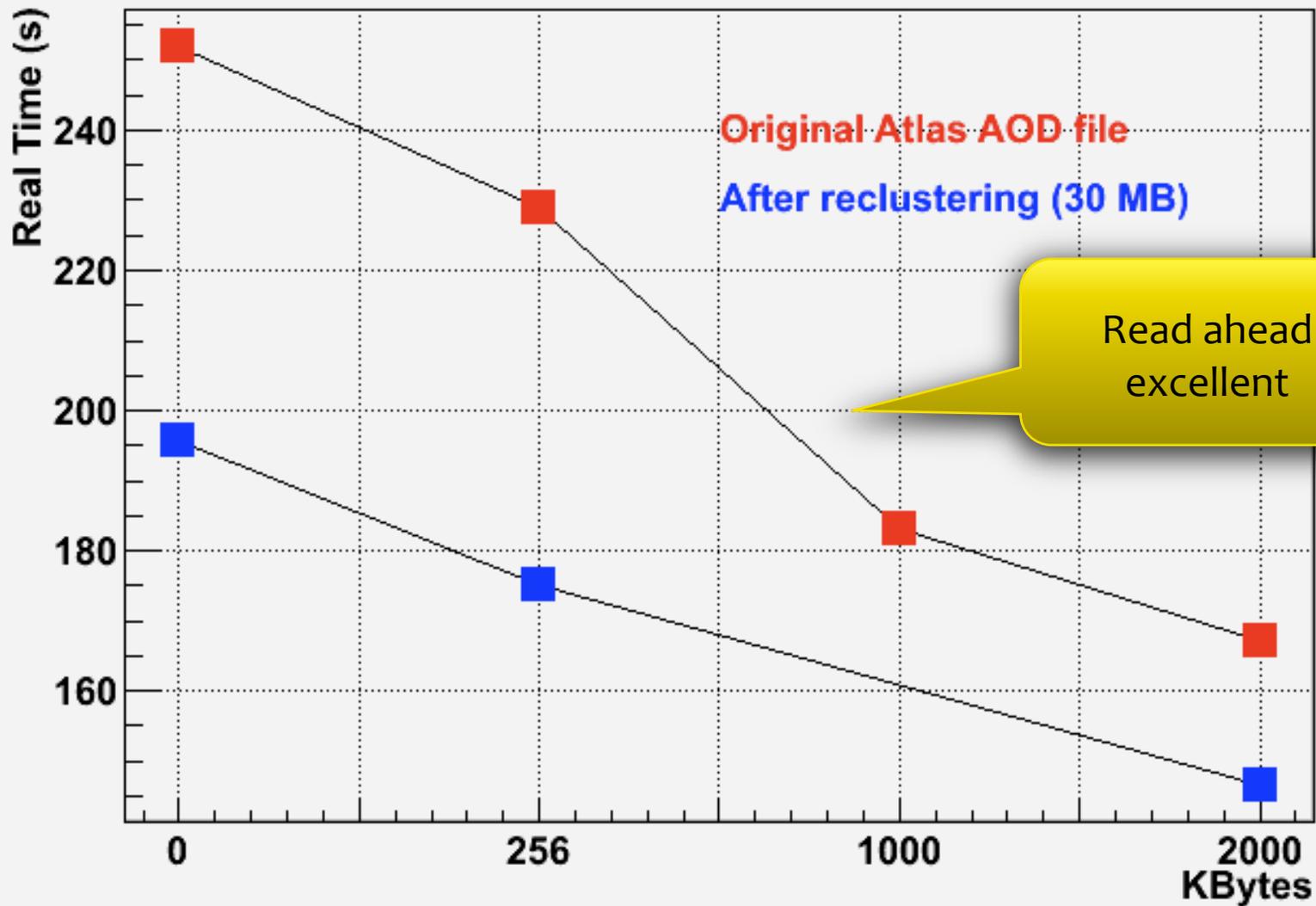
- The **readahead cache** will read all non consecutive blocks that are in the range of the cache.
- It minimizes the number of disk access. This operation could in principle be done by the OS, but the fact is that the OS parameters are not tuned for many small reads, in particular when many jobs read concurrently from the same disk.
- When using large values for the **TreeCache** or when the baskets are well sorted by entry, the **readahead cache** is not necessary.
- Typical (default value) is 256 Kbytes, although 2 Mbytes seems to give better results on Atlas files, but not with CMS or Alice.



Readahead

reading all branches, all entries

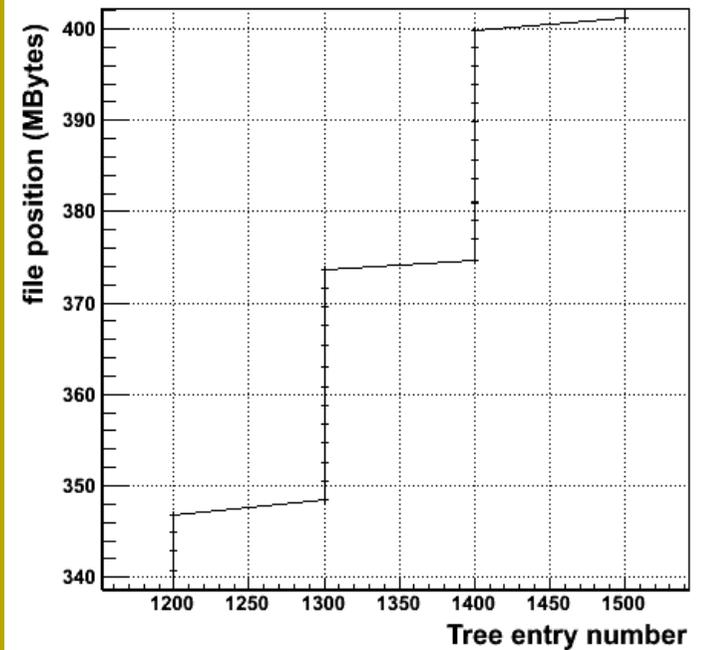
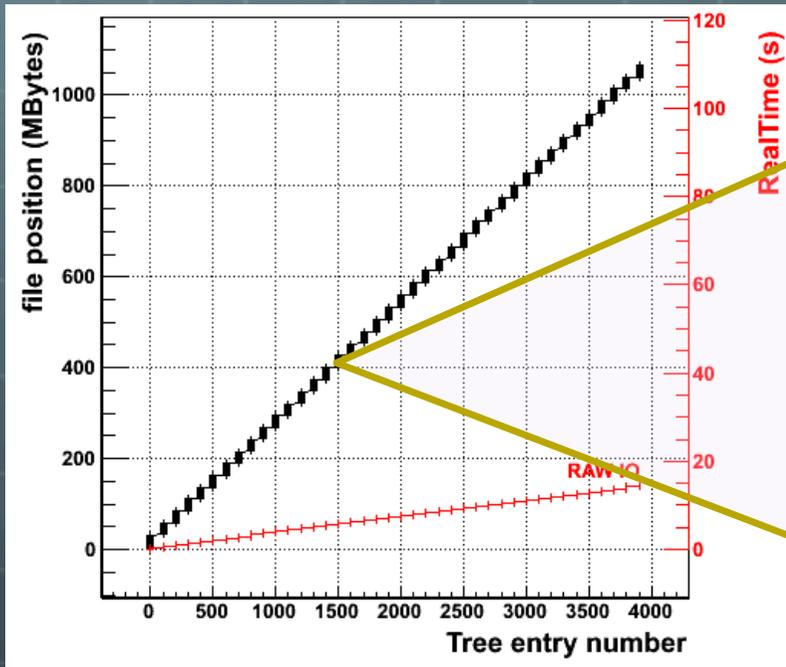
IO speed vs readahead cache size



OptimizeBaskets, AutoFlush

New in v5.25/04!

- 🌐 Solution, enabled by default:
- 🌐 Tweak basket size!
- 🌐 Flush baskets at regular intervals!



OptimizeBaskets

- 🌐 **Facts:** Users do not tune the branch buffer size
- 🌐 **Effect:** branches for the same event are scattered in the file.
- 🌐 **TTree::OptimizeBaskets** is a new function that will optimize the buffer sizes taking into account the population in each branch.
- 🌐 You can call this function on an existing read only Tree file to see the diagnostics.

FlushBaskets

- 🌐 **TTree::FlushBaskets** was introduced in 5.22 but called only once at the end of the filling process to disconnect the buffers from the tree header.
- 🌐 In version 5.25/04 this function is called automatically when a reasonable amount of data (default is 30 Mbytes) has been written to the file.
- 🌐 The frequency to call TTree::FlushBaskets can be changed by calling **TTree::SetAutoFlush**.
- 🌐 The first time that **FlushBaskets** is called, we also call **OptimizeBaskets**.

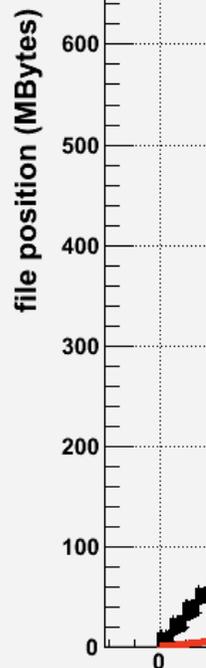
FlushBaskets 2

- 🌐 The frequency at which **FlushBaskets** is called is saved in the Tree (new member fAutoFlush).
- 🌐 This very important parameter is used when reading to compute the best value for the **TreeCache**.
- 🌐 The **TreeCache** is set to a multiple of fAutoFlush.
- 🌐 Thanks to **FlushBaskets** there is no backward seeks on the file for files written with 5.25/04. This makes a dramatic improvement in the raw disk IO speed.

Similar pattern with CMS files

RelValMinBias-GEN-SIM-RECO.root/Events

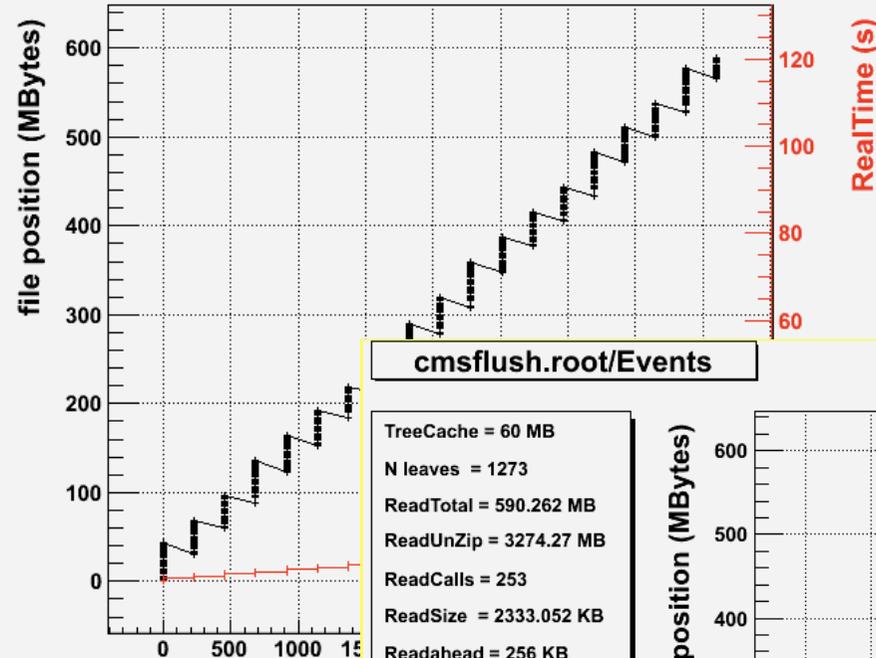
TreeCache = 0 MB
 N leaves = 1273
 ReadTotal = 566.008 MB
 ReadUnZip = 3295.41 MB
 ReadCalls = 143533
 ReadSize = 3.943 KB
 Readahead = 256 KB
 Readextra = 0.00 per cent
 Real Time = 130.853 s
 CPU Time = 111.280 s
 Disk Time = 20.899 s
 Disk IO = 27.084 MB/s
 ReadUZRT = 25.184 MB/s
 ReadUZCP = 29.614 MB/s
 ReadRT = 4.326 MB/s
 ReadCP = 5.086 MB/s



Darwin guest216.Inf.Root5.25/05, SVN :31431 Thu Nov 26 0

RelValMinBias-GEN-SIM-RECO.root/Events

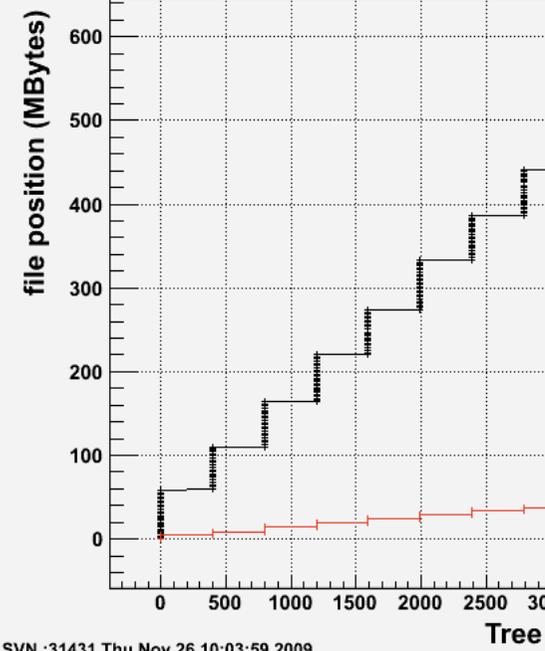
TreeCache = 30 MB
 N leaves = 1273
 ReadTotal = 566.008 MB
 ReadUnZip = 3295.41 MB
 ReadCalls = 996
 ReadSize = 568.281 KB
 Readahead = 256 KB
 Readextra = 7.02 per cent
 Real Time = 120.416 s
 CPU Time = 108.160 s
 Disk Time = 12.044 s
 Disk IO = 46.995 MB/s
 ReadUZRT = 27.367 MB/s
 ReadUZCP = 30.468 MB/s
 ReadRT = 4.700 MB/s
 ReadCP = 5.233 MB/s



Darwin macbrun2.cernRoot5.25/05, SVN :31472 Tue Dec 1 14:37:14 2009

cmsflush.root/Events

TreeCache = 60 MB
 N leaves = 1273
 ReadTotal = 590.262 MB
 ReadUnZip = 3274.27 MB
 ReadCalls = 253
 ReadSize = 2333.052 KB
 Readahead = 256 KB
 Readextra = 0.00 per cent
 Real Time = 120.530 s
 CPU Time = 113.650 s
 Disk Time = 10.130 s
 Disk IO = 58.269 MB/s
 ReadUZRT = 27.166 MB/s
 ReadUZCP = 28.810 MB/s
 ReadRT = 4.897 MB/s
 ReadCP = 5.194 MB/s



Darwin guest216.Inf.Root5.25/05, SVN :31431 Thu Nov 26 10:03:59 2009

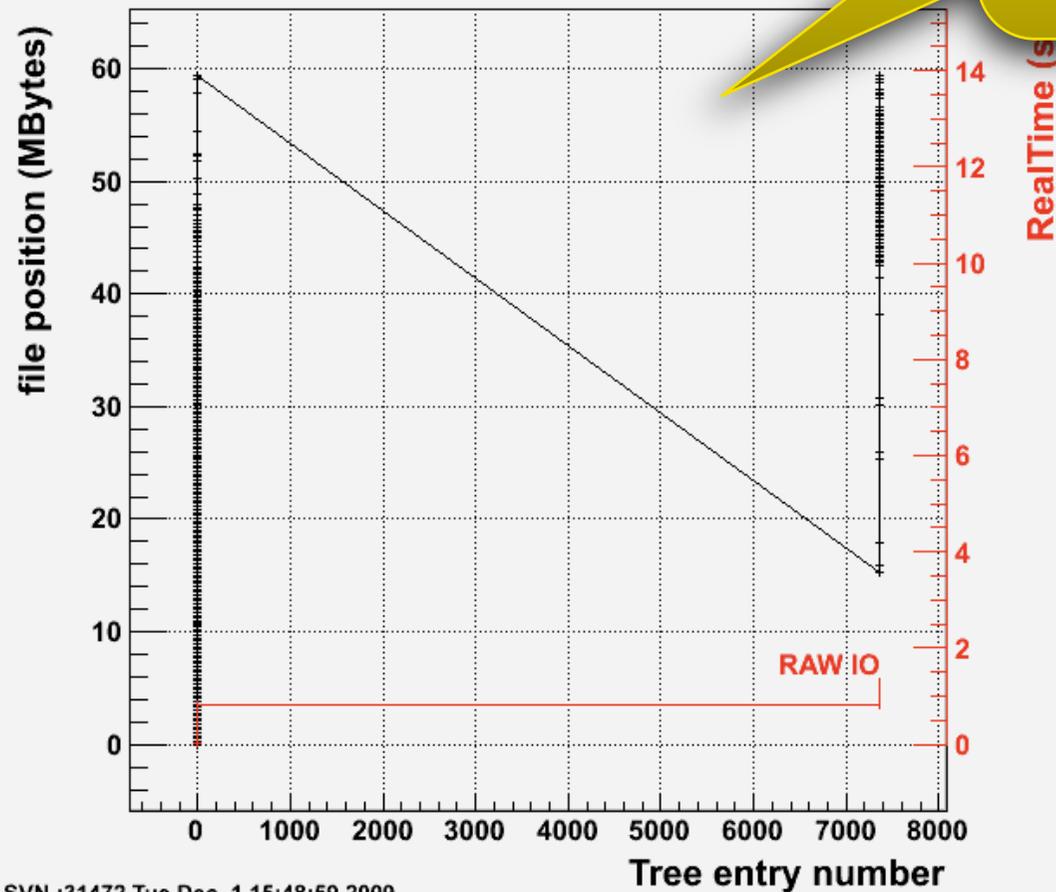
CMS : mainly CPU problem
 due to a complex object
 model

Alice files

Only small files used
in the test.
Performance
improvement with
5.26

RAW/AlIESDs.root/esdTree

TreeCache = 30 MB
N leaves = 593
ReadTotal = 29.7852 MB
ReadUnZip = 1755.39 MB
ReadCalls = 269
ReadSize = 110.726 KB
Readahead = 256 KB
Readextra = 104.41 per cent
Real Time = 13.884 s
CPU Time = 11.320 s
Disk Time = 1.300 s
Disk IO = 22.904 MB/s
ReadUZRT = 126.431 MB/s
ReadUZCP = 155.070 MB/s
ReadRT = 2.145 MB/s
ReadCP = 2.631 MB/s



LHCb files

Performance
must be very
poor !!

- 🌐 One LHCb file contains about 45 Trees !!
- 🌐 Each Tree should be a top level branch in the main Tree
 - 🌐 To take advantage of the **TreeCache**
 - 🌐 To take advantage of **FlushBaskets**

TreeCache

Interface for
Different use patterns

Use Case 1

Reading all branches

```
void taodr(Int_t cachesize=10000000) {
    TFile *f = TFile::Open("AOD.067184.big.pool.root");
    TTree *T = (TTree*)f->Get("CollectionTree");
    Long64_t nentries = T->GetEntries();
    T->SetCacheSize(cachesize);
    T->AddBranchToCache("*", kTRUE);

    for (Long64_t i=0; i<nentries; i++) {
        T->GetEntry(i);
    }
}
```

Use Case 2

Reading only a few branches in consecutive events

```
void taodr(Int_t cachesize=10000000) {
    TFile *f = TFile::Open("AOD.067184.big.pool.root");
    TTree *T = (TTree*)f->Get("CollectionTree");
    Long64_t nentries=1000;
    Long64_t efirst= 2000;
    Long64_t elast = efirst+nentries;
    T->SetCacheSize(cachesize);
    T->SetCacheEntryRange(efirst,elast);
    TBranch *b_m_trigger_info = T->GetBranch("m_trigger_info");
    TBranch *b_m_compactData = T->GetBranch("m_compactData");
    T->AddBranchToCache(b_m_trigger_info,kTRUE);
    T->AddBranchToCache(b_m_compactData, kTRUE);
    T->StopCacheLearningPhase();

    for (Long64_t i=0;i<nentries;i++) {
        T->LoadTree(i);
        b_m_trigger_info->GetEntry(i);
        b_m_compactData->GetEntry(i);
    }
}
```

Use Case 2 results

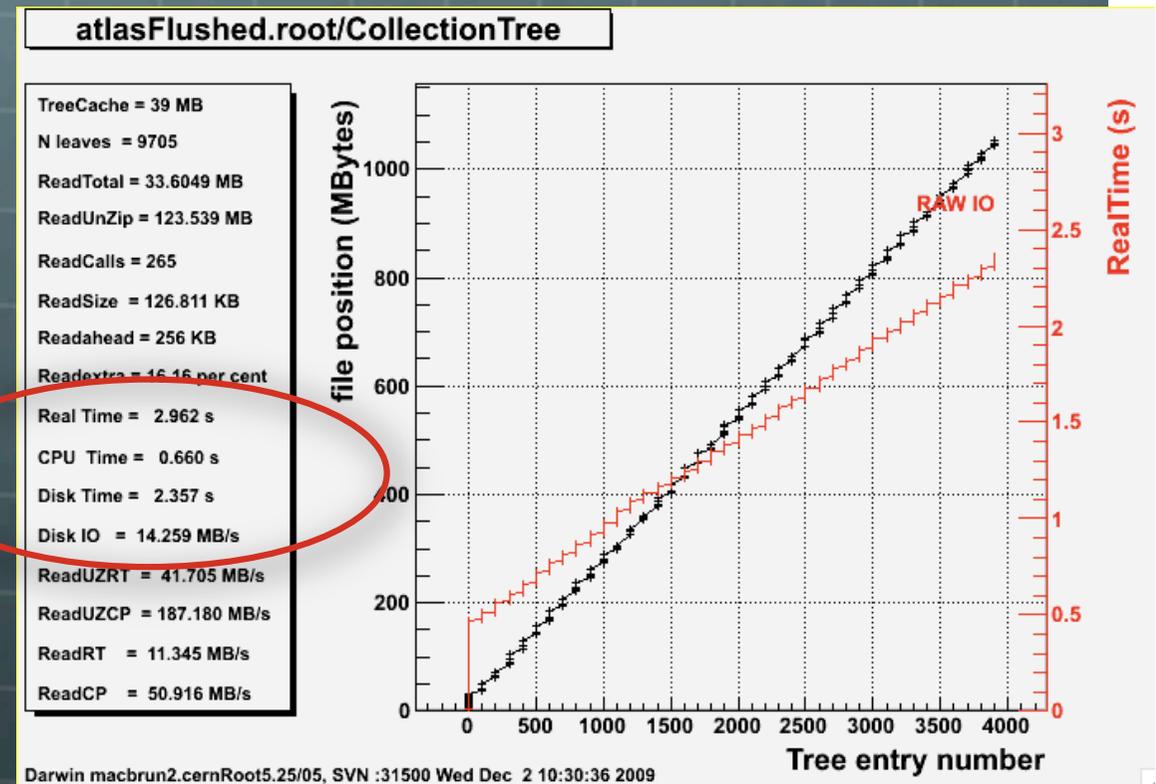
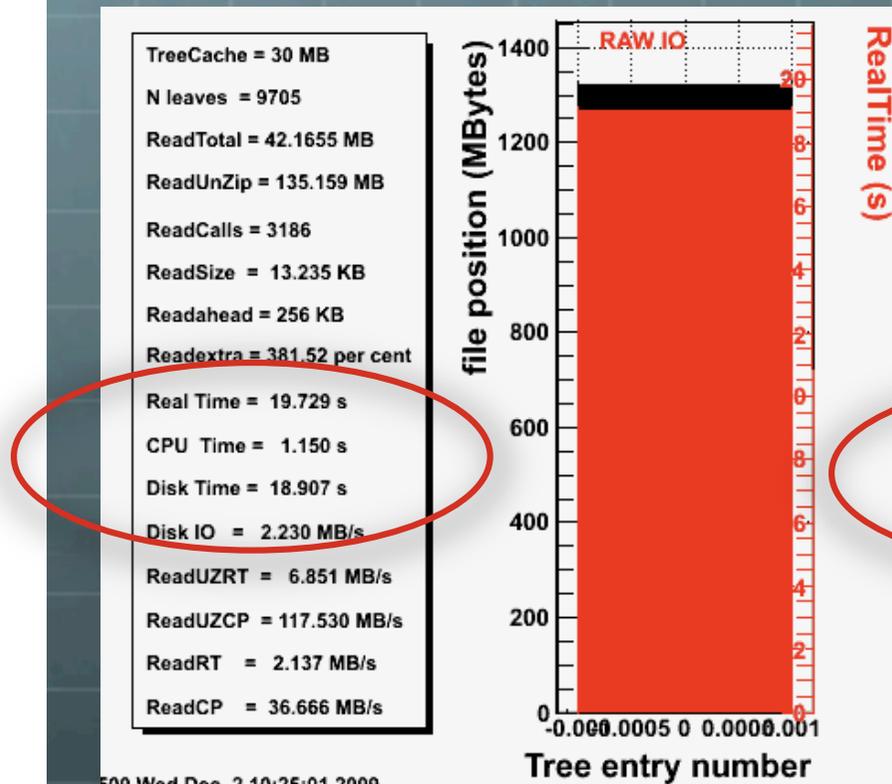
reading 33 Mbytes out of 1100 MBytes

Seek time = $3186 * 5ms = 15.9s$

Seek time = $265 * 5ms = 1.3s$

Old ATLAS file

New ATLAS file



Use Case 3

Select events with one branch
then read more branches

```
void taodr(Int_t cachesize=10000000) {
    TFile *f = TFile::Open("AOD.067184.big.pool.root");
    TTree *T = (TTree*)f->Get("CollectionTree");
    Long64_t nentries=1000;
    Long64_t efirst= 2000;
    Long64_t elast = efirst+nentries;
    T->SetCacheSize(cachesize);
    T->SetCacheEntryRange(efirst,elast);
    T->SetCacheLearnEntries(10);
    TBranch *b_m_trigger_info = T->GetBranch("m_trigger_info");
    T->AddBranchToCache(b_m_trigger_info,kTRUE);

    for (Long64_t i=0;i<nentries;i++) {
        T->LoadTree(i);
        b_m_trigger_info->GetEntry(i);
        if (somecondition) readmorebranches();
    }
}
```

All branches used
in the first 10
entries will be
cached

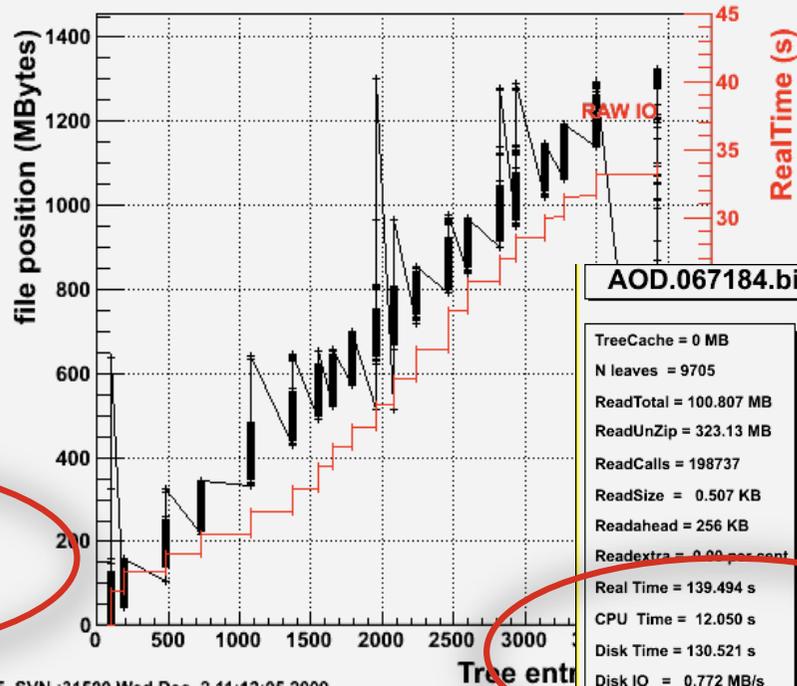
Use Case 3 results

reading 1% of the events

Even in this difficult case cache is better

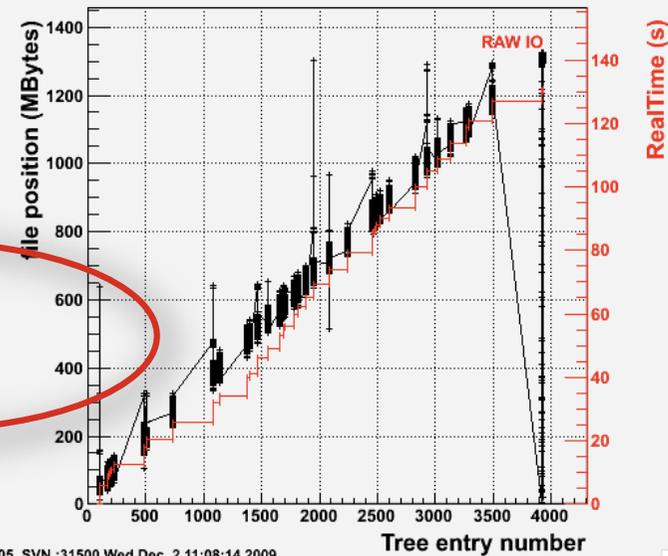
AOD.067184.big.pool_4.root/CollectionTree

TreeCache = 30 MB
 N leaves = 9705
 ReadTotal = 683.905 MB
 ReadUnZip = 2192.22 MB
 ReadCalls = 6895
 ReadSize = 99.189 KB
 Readahead = 256 KB
 Readextra = 92.61 per cent
 Real Time = 40.916 s
 CPU Time = 8.830 s
 Disk Time = 33.823 s
 Disk IO = 20.220 MB/s
 ReadUZRT = 53.579 MB/s
 ReadUZCP = 248.270 MB/s
 ReadRT = 16.715 MB/s
 ReadCP = 77.452 MB/s



AOD.067184.big.pool_3.root/CollectionTree

TreeCache = 0 MB
 N leaves = 9705
 ReadTotal = 100.807 MB
 ReadUnZip = 323.13 MB
 ReadCalls = 198737
 ReadSize = 0.507 KB
 Readahead = 256 KB
 Readextra = 0.00 per cent
 Real Time = 139.494 s
 CPU Time = 12.050 s
 Disk Time = 130.521 s
 Disk IO = 0.772 MB/s
 ReadUZRT = 2.316 MB/s
 ReadUZCP = 26.816 MB/s
 ReadRT = 0.723 MB/s
 ReadCP = 8.366 MB/s



Darwin macbrun2.cernRoot5.25/05, SVN :31500 Wed Dec 2 11:12:05 2009

Darwin macbrun2.cernRoot5.25/05, SVN :31500 Wed Dec 2 11:08:14 2009

Use Case 3 results

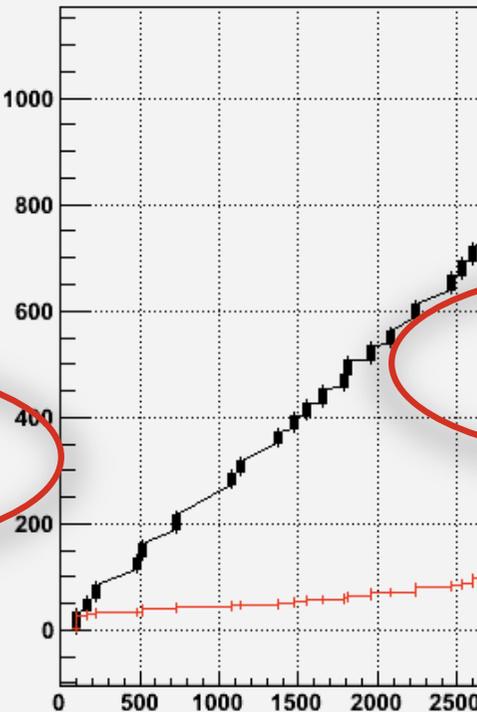
reading 1% of the events

Even in this difficult case cache is better

atlasFlushed.root/CollectionTree

TreeCache = 39 MB
 N leaves = 9705
 ReadTotal = 692.292 MB
 ReadUnZip = 2545.01 MB
 ReadCalls = 346
 ReadSize = 2000.844 KB
 Readahead = 256 KB
 Readextra = 0.00 per cent
 Real Time = 19.310 s
 CPU Time = 17.590 s
 Disk Time = 2.222 s
 Disk IO = 311.600 MB/s
 ReadUZRT = 131.800 MB/s
 ReadUZCP = 144.685 MB/s
 ReadRT = 35.852 MB/s
 ReadCP = 39.357 MB/s

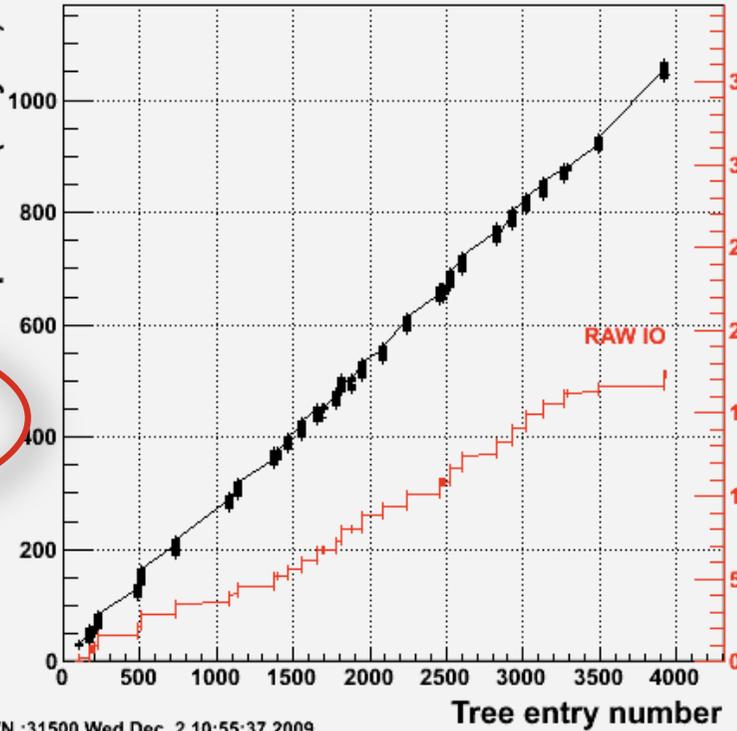
file position (MBytes)



atlasFlushed.root/CollectionTree

TreeCache = 0 MB
 N leaves = 9705
 ReadTotal = 604.979 MB
 ReadUnZip = 2224.03 MB
 ReadCalls = 263853
 ReadSize = 2.293 KB
 Readahead = 256 KB
 Readextra = 0.00 per cent
 Real Time = 35.367 s
 CPU Time = 20.230 s
 Disk Time = 17.376 s
 Disk IO = 34.816 MB/s
 ReadUZRT = 62.885 MB/s
 ReadUZCP = 109.937 MB/s
 ReadRT = 17.106 MB/s
 ReadCP = 29.905 MB/s

file position (MBytes)



RealTime (s)

Darwin macbrun2.cernRoot5.25/05, SVN :31500 Wed Dec 2 10:55:37 2009

Darwin macbrun2.cernRoot5.25/05, SVN :31500 Wed Dec 2 10:59:56 2009

Tree entry number

Reading network files

Same results
with xrootd

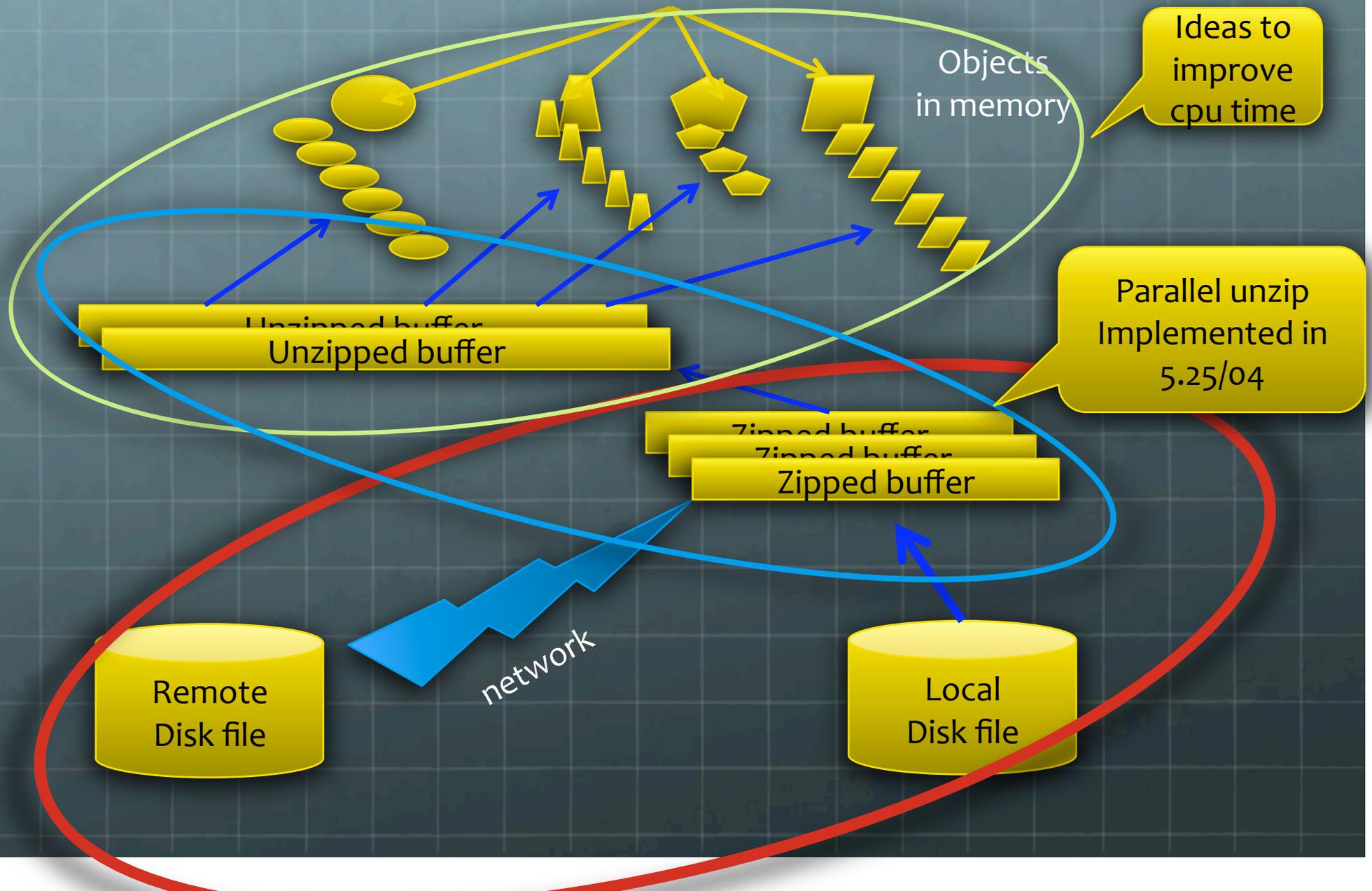
```
f = TFile::Open("http://root.cern.ch/files/AOD.067184.big.pool_4.root")  
f = TFile::Open("http://root.cern.ch/files/atlasFlushed.root")
```

File type Cache on/off	Local file	LAN = 0.3ms 1 Gbit/s	WLAN = 3ms 100 Mbits/s	ADSL=72ms 8 Mbits/s
Atlas orig CA=OFF	RT=720s TR=1328587	RT=1156s NT=400+12s	RT=4723s NT=4000+120s	RT > 2 days NT>2 days
Atlas orig CA=ON	RT=132s TR=323	RT=144s NT=0.1+12s	RT=253s NT=1+120s	RT=1575s NT=223+1200s
Atlas flushed CA=OFF	RT=354s TR=486931	RT=485s NT=120+11s	RT=1684 NT=1200+110s	RT= NT>1 day
Atlas flushed CA=ON	RT=106s TR=45	RT=117s NT=0.03+11s	RT=217s NT=0.1+110s	RT=1231s NT=25+1100s

TR=Transactions

NT=Network Time (latency + data transfer)

Important factors



Summary

- 🌐 Use of the **TreeCache** is essential, even with local files
- 🌐 Flushing buffers at regular intervals help a lot
- 🌐 In ROOT 5.25/04, **FlushBaskets** is ON by default and also an automatic basket buffer optimization.
- 🌐 In ROOT 5.25/04 a simplified interface to TreeCache is provided via TTree.
- 🌐 Files written with 5.25/04 will be faster to read

Summary 2

- 🌐 Version 5.26 will be released on December 15.
- 🌐 This version is back compatible with 5.22 and 5.24.
- 🌐 We hope that collaborations will move soon to this new version.