



# Performance Improvements with ATLAS AOD files

Rene Brun  
3 November 2009

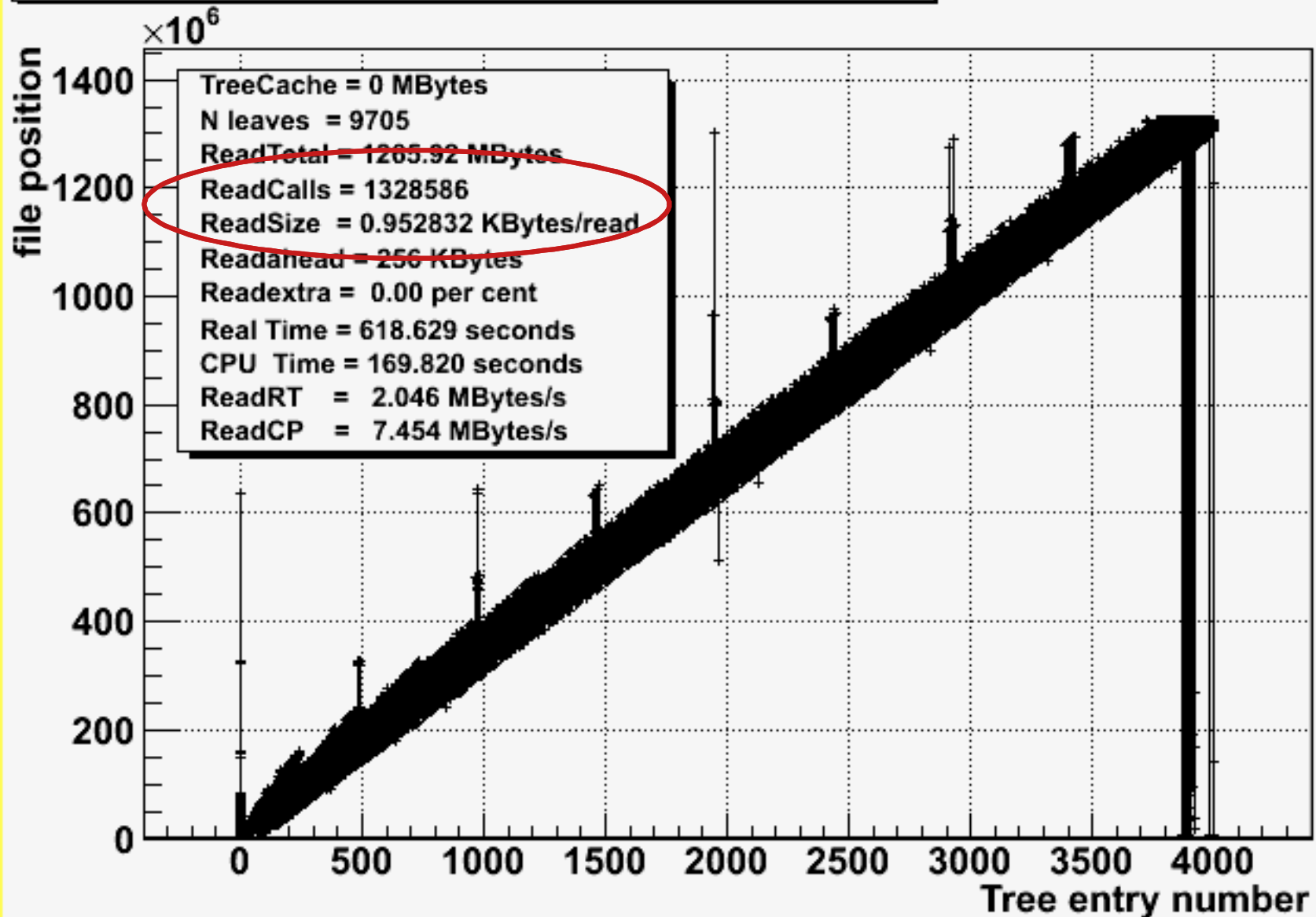


# Main Points

- Typical problems with Trees
  - Branch buffers not clustered by entry
  - Forward/backward seeks when reading
  - Too many network transactions
  - Expensive object model (cpu time)
- Solutions
  - **TTreeCache**
  - **Readahead** buffer
  - Reclustering online or a-posteriori
  - Cheaper object model
  - Monitoring with **TTreePerfStats**

# See Doctor

AOD.067184.big.pool.root/CollectionTree





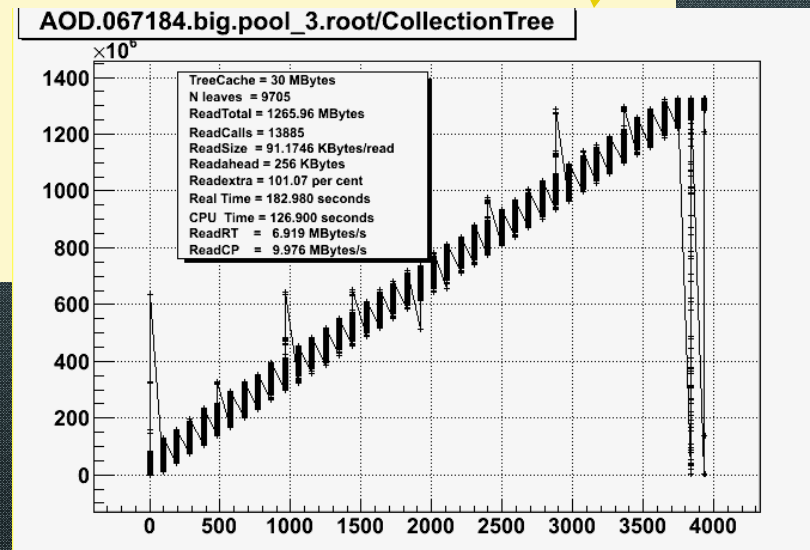
# Use TTreePerfStats

```
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);
    if (cachesize > 0) {
        T->SetCacheEntryRange(0, nentries);
        T->AddBranchToCache("*", kTRUE);
    }

    TTreePerfStats ps("ioperf", T);

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

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



# 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

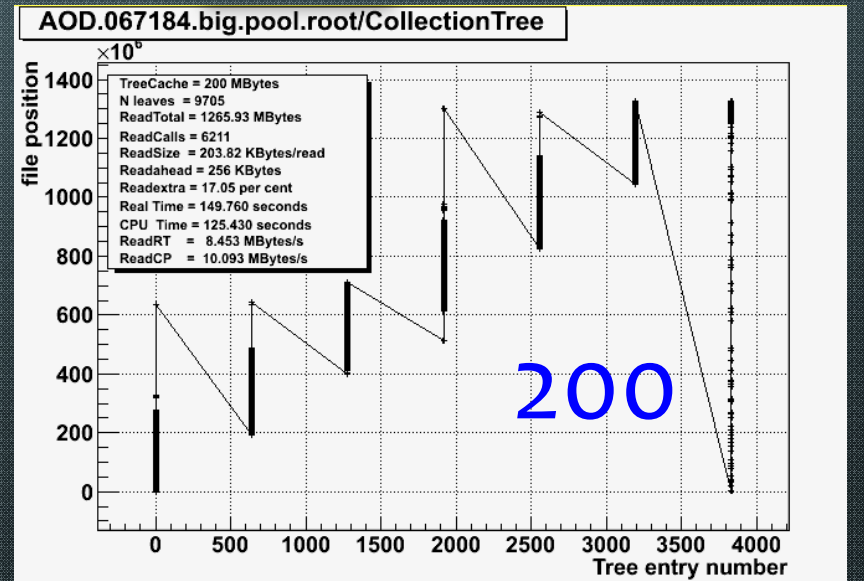
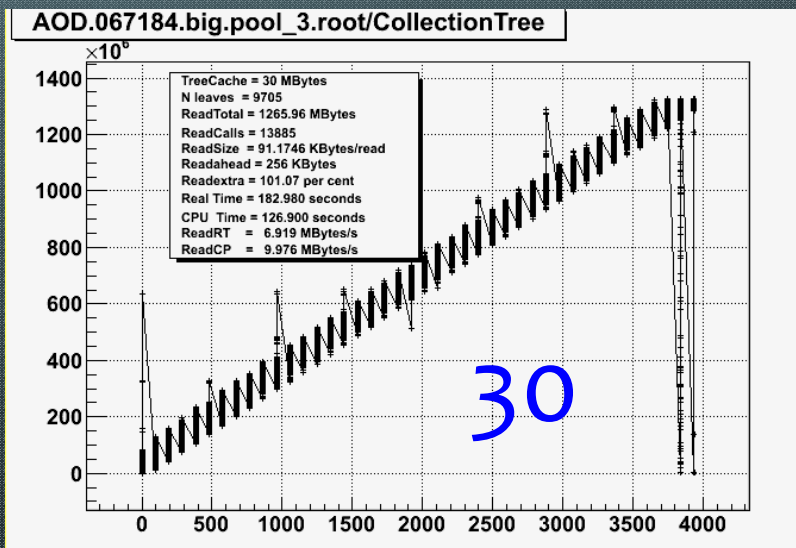
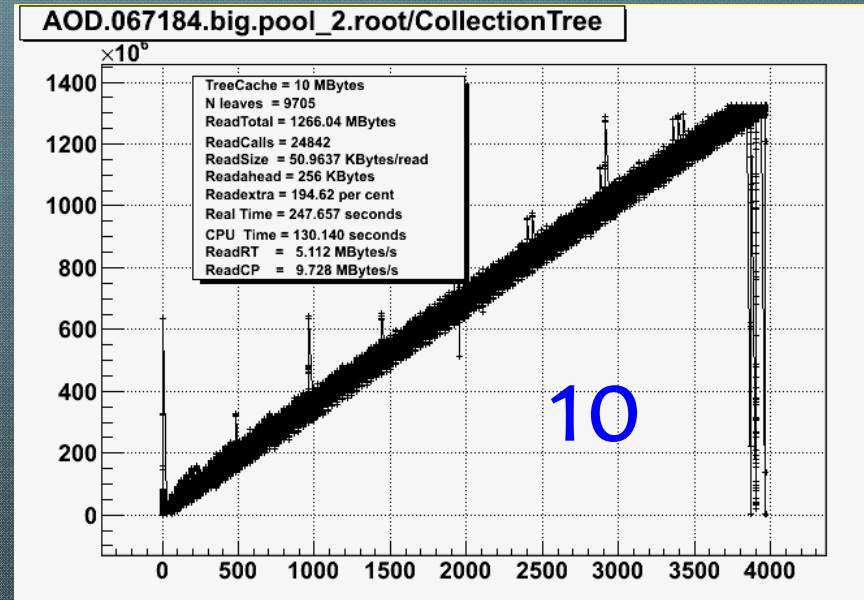
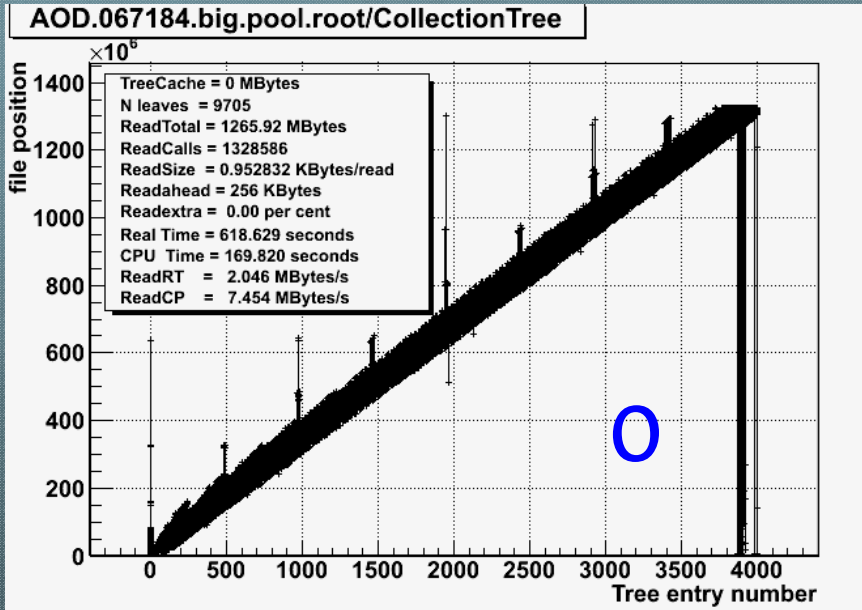
- 🌐 Using one of the AOD files the class headers have been generated automatically via `TTree::MakeProject`.
- 🌐 The corresponding shared library is linked such that the same object model is used in my tests and in Atlas persistent model.
- 🌐 The tests read systematically all entries in all branches. 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.



# What is the TreeCache

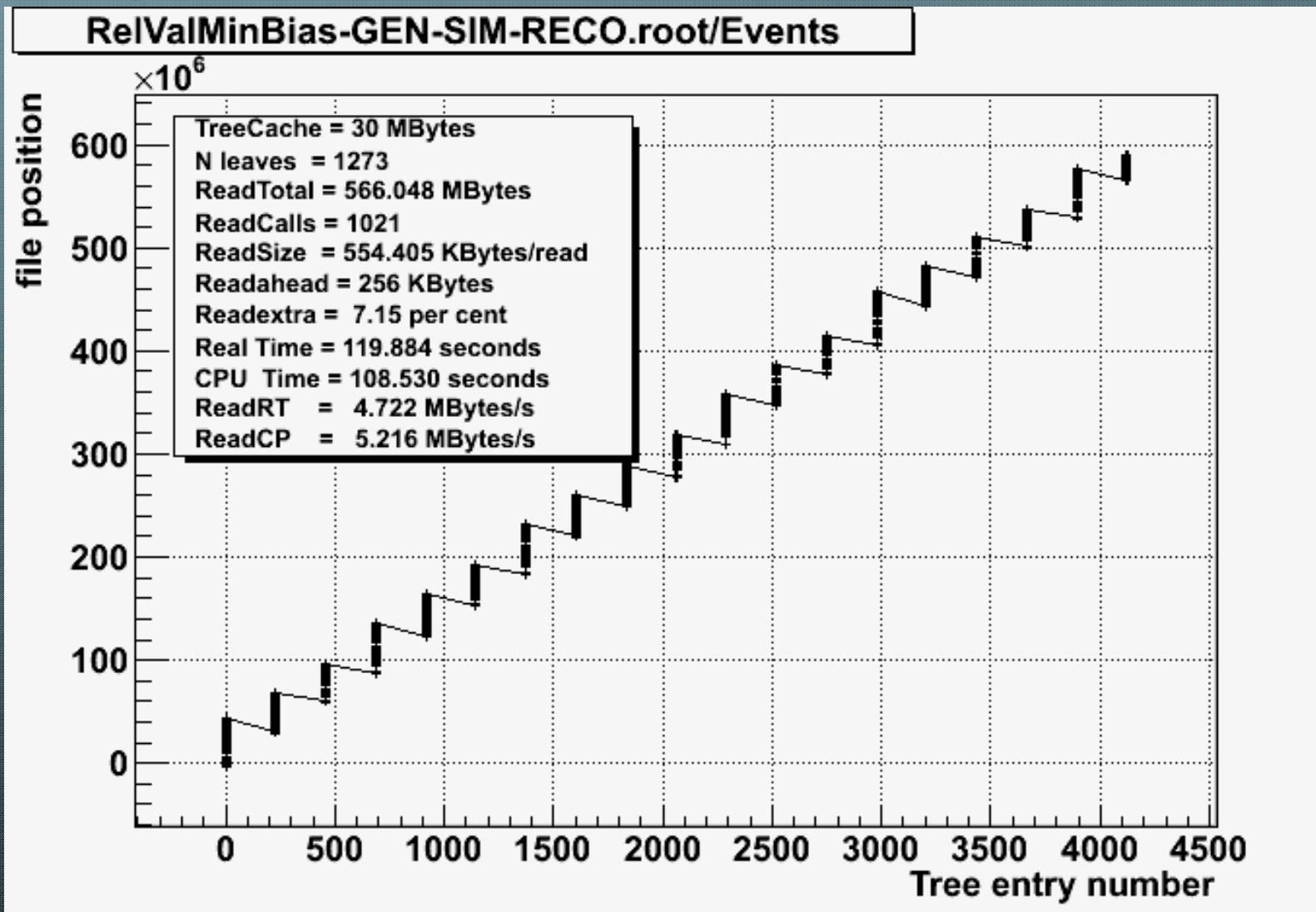
-  It groups into one buffer all blocks from the used branches.
-  The blocks are sorted in ascending order and consecutive blocks merged such that the file is read sequentially.
-  It reduces typically by a factor 1000 the number of transactions with the disk and in particular the network with servers 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 values. If you have no memory problem, set large values like 200 Mbytes.

# TreeCache size impact



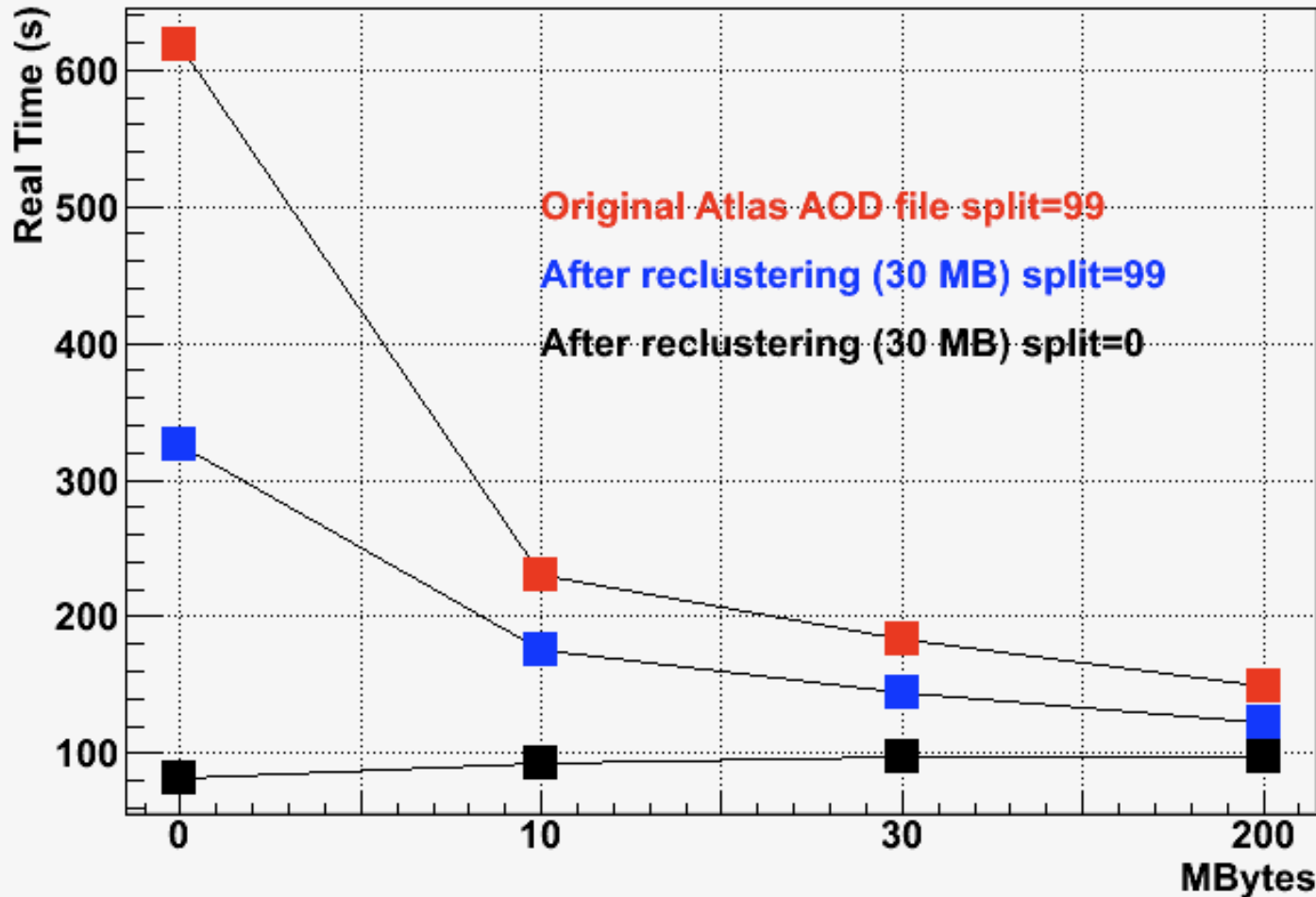


# Similar pattern with CMS files



# TreeCache results graph

IO speed vs TreeCache size





# 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

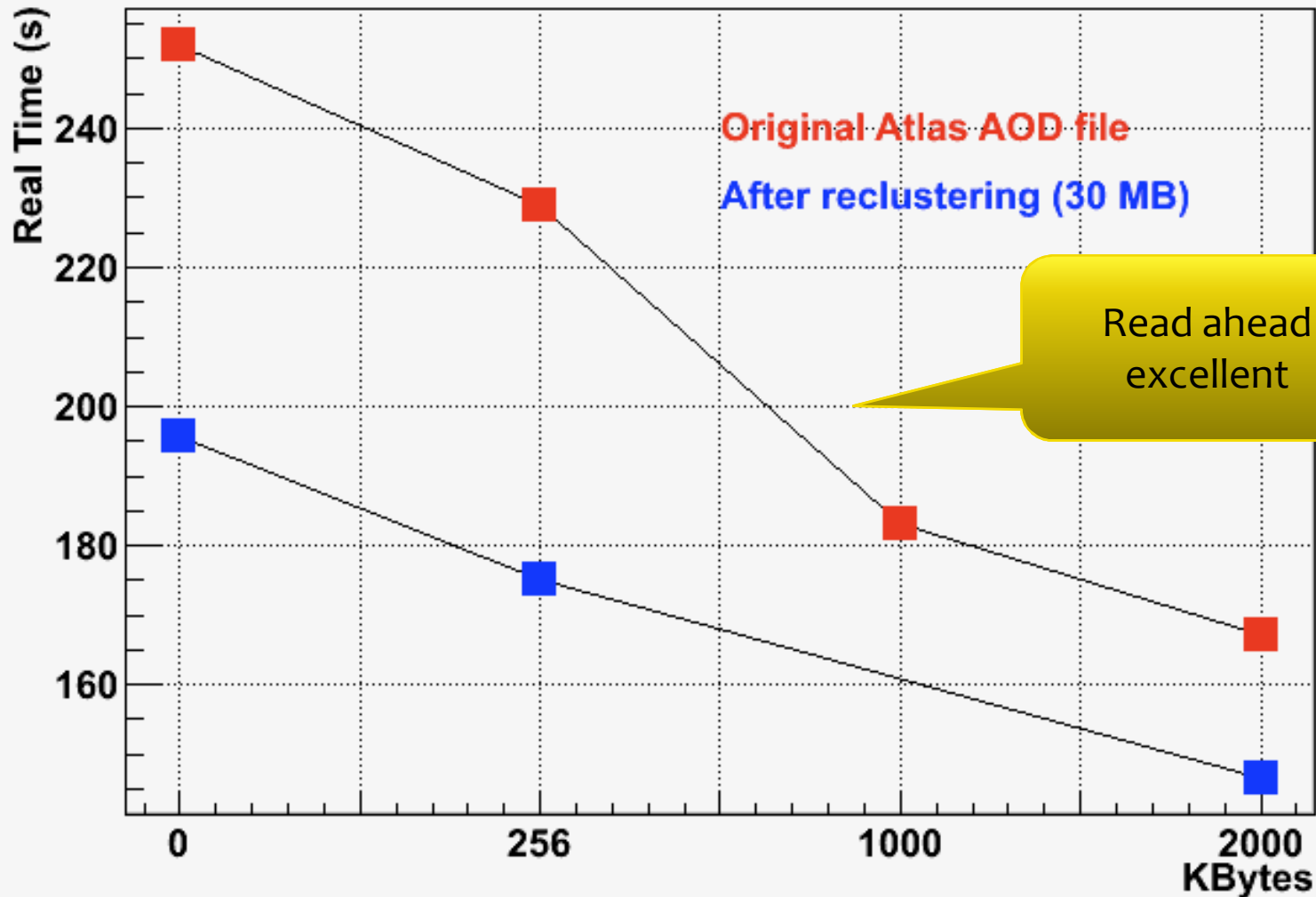
# What is the readahead cache

- 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.
- The readahead cache should not be used in several use cases (see 2 examples later)

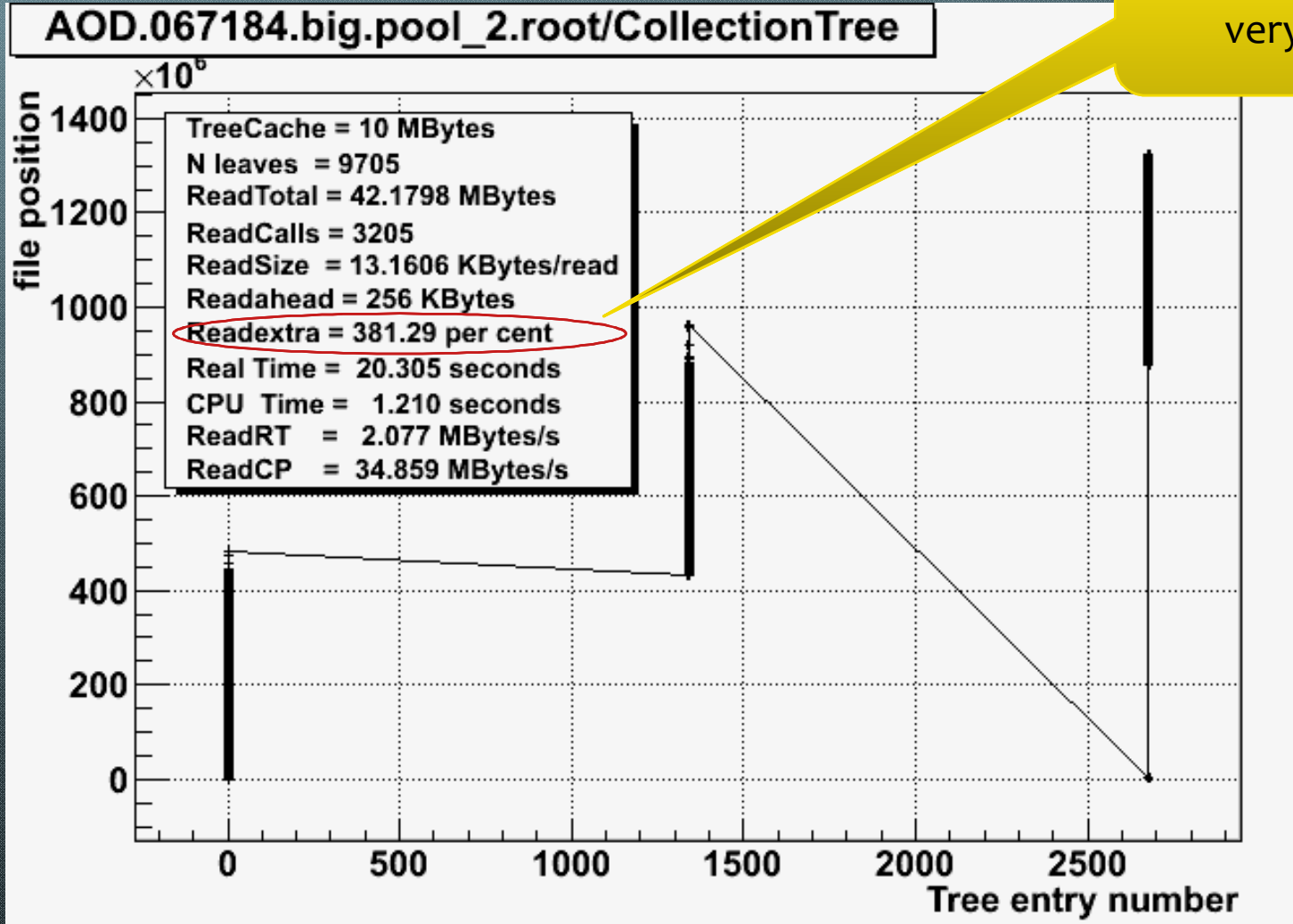


# Readahead reading all branches, all entries

IO speed vs readahead cache size



# Reading only 2 branches out of 9705



Read ahead very bad



# Reading all branches in 1% random entries

Read ahead  
very bad

AOD.067184.big.pool\_4.root/CollectionTree

