Improving performance of AliRoot using C++11

I had started with learning about what is ROOT and how are things defined in it. It took some time getting acquainted with ROOT. I moved on to seeing things of C++11 in a quick grasp. All this just created a base for me to work on my project. The next step from here on which I took was profiling the example tests to see how functions are being called out and what is taking in more time of processing of the examples so that I can shift my focus on to the process of modifying and improving it. The profiling results showed that there was a much use of TobjectArray and TclonesArray so we wanted to see if we can replace them using C++ standard libraries.

There were three ways of replacing TobjectArray using C++11 standard libraries and they were

1. **Vector of object pointers**
2. **Vector of unique pointers of object**
3. **Vector of shared pointers of object**

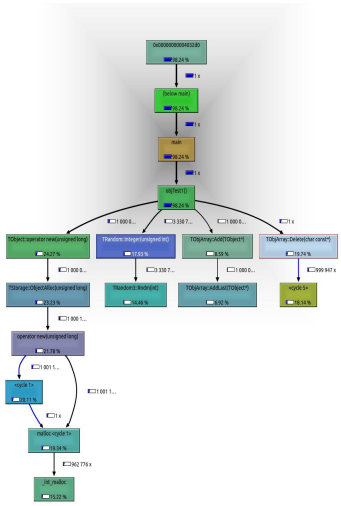
Before replacing it, we should know how much difference does it make so to see that I had defined a base class(MyClass – which is a general shape class) and three derived classes(Rectangle, Right angle triangle and Cuboid) of it.

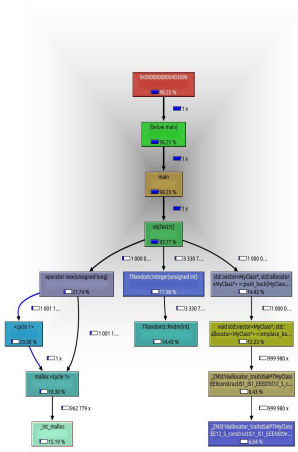
The results of comparison were as follows:
Memory Access::
First, I had started generating randomly derived classes of it storing into the respective datatype(10,000,000) and using the datatype to call the functions of the classes.
So the performance of different implementations are::
Number of entries:: 10,000,000
Using vector<unique_ptr> to store it :: 7.43752
Using vector<shared_ptr> to store it :: 6.23267
Using vector<myclass*> to store it :: 2.54501
Using TObject array to store it :: 2.62404

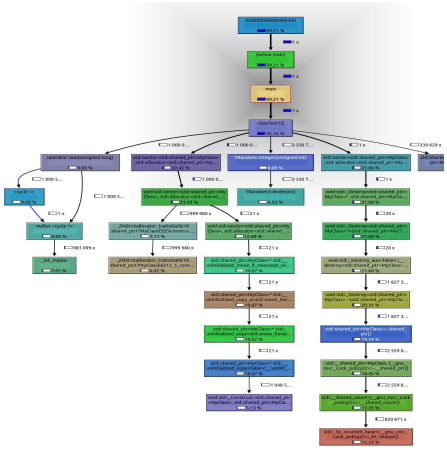
Read and Write(+Memory access)::
This had been done to compare the performance of storing a collection of these objects in a TObject array and a vector of base class pointers.
So the results for reading and writing it in a root file using a tree structure are::
For writing,
Number of entries:: 1,000,000
Time taken with TObjectArr:: 1.62437
Time taken with vector<myclass*>:: 1.48291

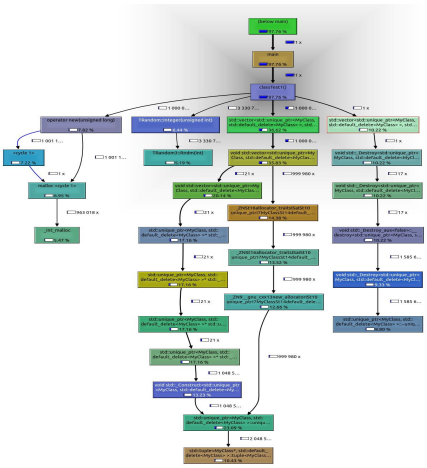
For reading,
Time taken with TObjectArr:: 1.16272
Time taken with vector<myclass*>:: 0.928012

The profiling results of the respective datatypes used are shown below:

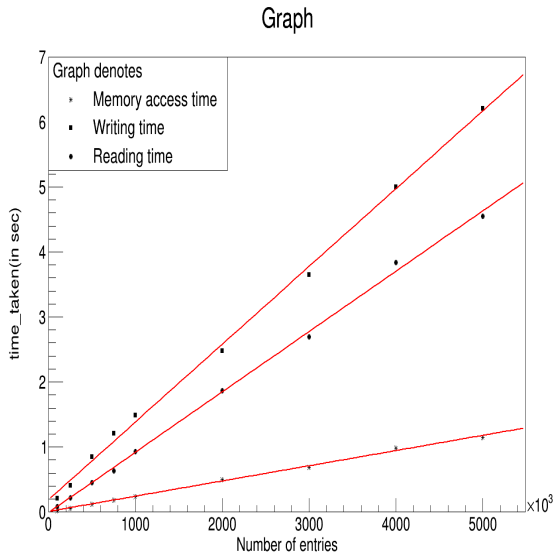


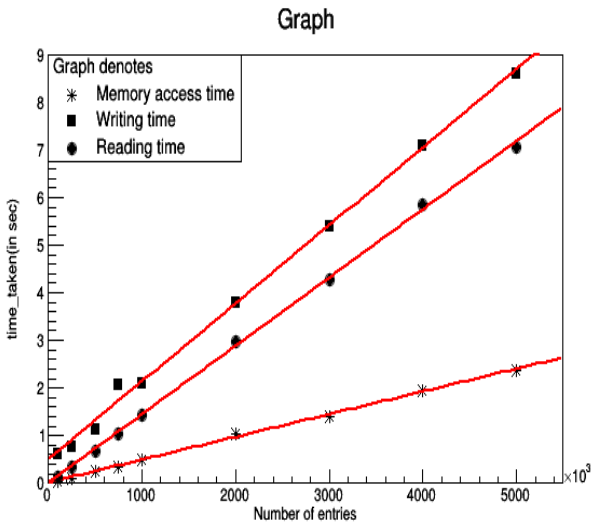






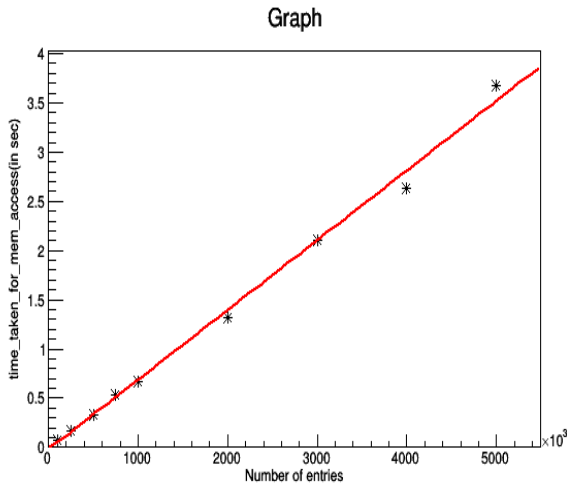
The graphs showing the time with number of entries had also been plotted and they are as shown below:



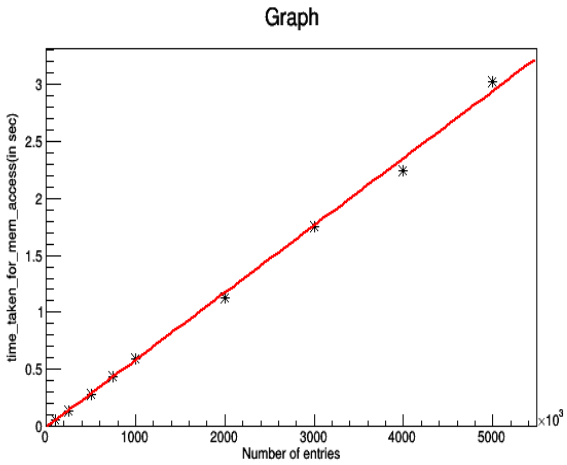


vector<pointers>

TobjectArray



vector<unique_pointers>

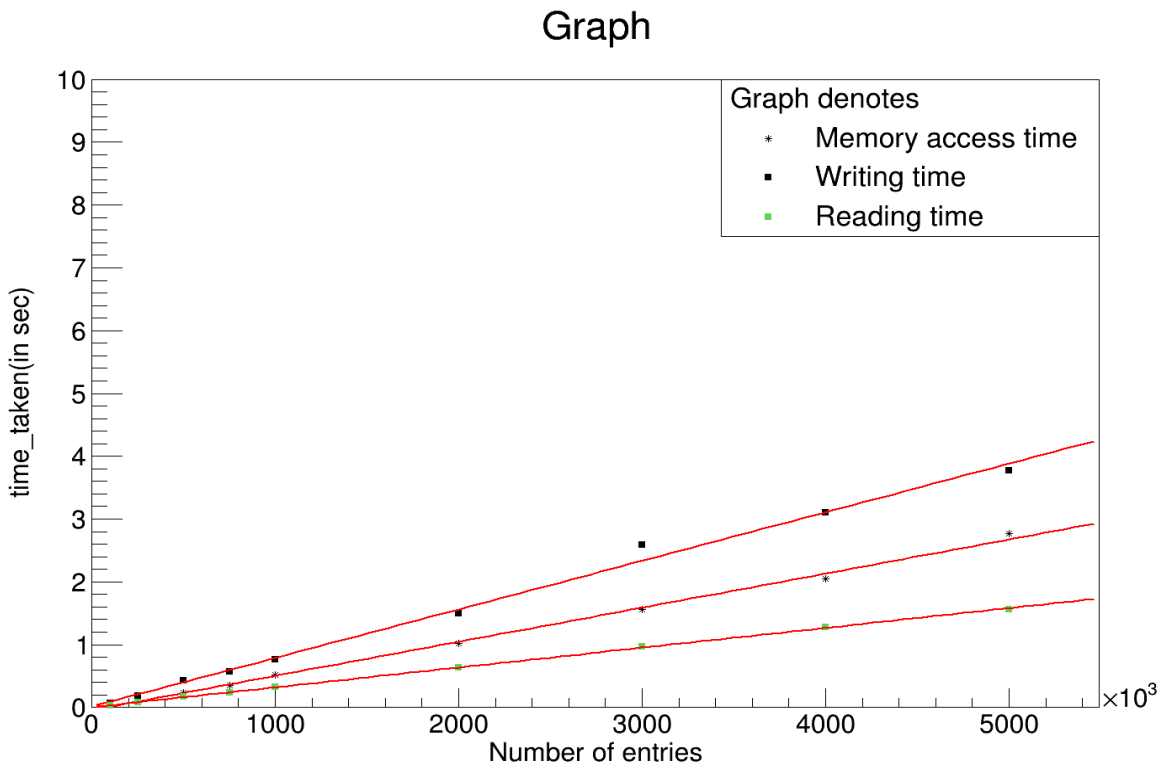


vector<shared_pointers>

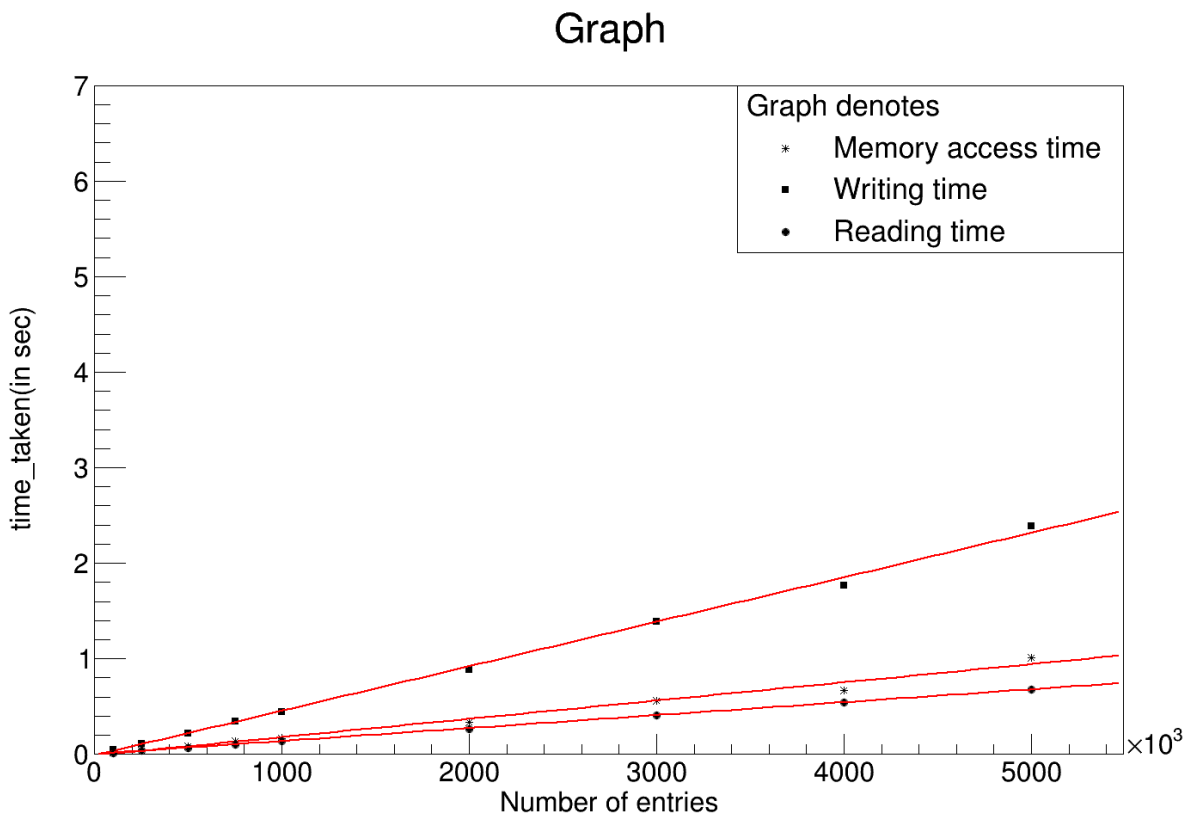
TclonesArray vs Vector<objects>

After comparing this, we moved on to compare TclonesAraay with vector of objects.
Comparing TClonesArray and vector of a derived class for performance of read and write operations on a TFile using a TTree.
So the results for reading and writing are::
Number of entries:: 1,000,000
For writing,
Time taken with TClonesArray:: 0.622415
Time taken with vector<derived>:: 0.520319

For reading,
Time taken with TClonesArray:: 0.247496
Time taken with vector<derived>:: 0.135303



TClonesArray



vector<objects>

After seeing a huge improvement when comparing TclonesArray with vector of objects, we had planned to try and replace the TclonesArray with vector implentation without changing any functionality of TclonesArray but there were many problems with it. One of the major one was defining a vector after taking the input from user for what type of datatype does the vector should be of. We could not do this so we have moved on to add a new implementation to ROOT.