

ANN/Objectivity

Status report and performance test on the “re-clustered” tag DB

Gabriele Garzoglio, Apr. 13 2001

This document reports the status of development of ANN/objectivity and the tests performed on TAM (Terabyte Analysis Machine)¹ on the re-clustering module. This module speeds up query search time by 15-20% on databases of millions of objects.

ANN/Objectivity is a library used to solve the problem of the k^{th} nearest neighbors in large set of data stored in objectivity databases. The main goal of the project was providing an analysis tool capable of dealing with typical data sizes at least 2 order of magnitude larger than what can fit into memory (hundreds of Gbytes). Once achieved this goal, it was pretty clear that just reorganizing the layout of the data on disk could somewhat improve the search speed (re-clustering module). So far, this library has being used to analyze the Sloan Digital Sky Survey data.

To date, ANN/Objectivity makes use of a utility called the “Tag extractor” to create an objectivity database of flat tag objects: the search for the k^{th} nearest neighbors is then performed on this database. This utility, written by Koen Holtman (Caltech) and modified by myself, selects the `sxPhotoTag` objects from the SDSS objectivity database (SX, EDR v2.2.1)², eliminates the internal references to the other SX objects (make it “flat”, stand alone) and writes them into a new objectivity federation. The tag extractor can be configured to extract only tags from certain kind of objects (e.g. from the galaxy and/or the star containers).

The time to extract the 332 Bytes flat tags from the 6,650,927 galaxies of the SX database was 1h 18m, reading/writing at an average bandwidth of 470 Kbytes/s.

ANN searches the nearest neighbors organizing the data in a tree structure. This process introduces a natural indexing on the data. The data can then be reorganized on disk according to this indexing (re-clustering). This technique aims to minimize the amount of memory cache swaps needed by objectivity during the search for the nearest neighbors.

Figure 1 shows the amount of time per object required to build a kd-tree³ (4 dimensions, bucket size of 10) and re-cluster a database, for different database sizes (expressed in number of object). As expected, the complexity of the algorithm that builds the tree is $O(n \log(n))^4$ and the time to re-cluster the database is linear. It is to be noted that the calculated tree can be saved and retrieved at a rate of about 10 us per object, which is negligible with respect to nearest neighbors search time.

Figure 2 compares the time per object required to search 5 nearest neighbors for all the objects, in the two cases of standard and re-clustered database. For a small database (500,000 objects) the time is approximately the same: the data size in this case is small (160MB) and few memory swaps are needed; for bigger databases (above 2,500,000 objects) there is a net gain of 15-20% in the query execution.

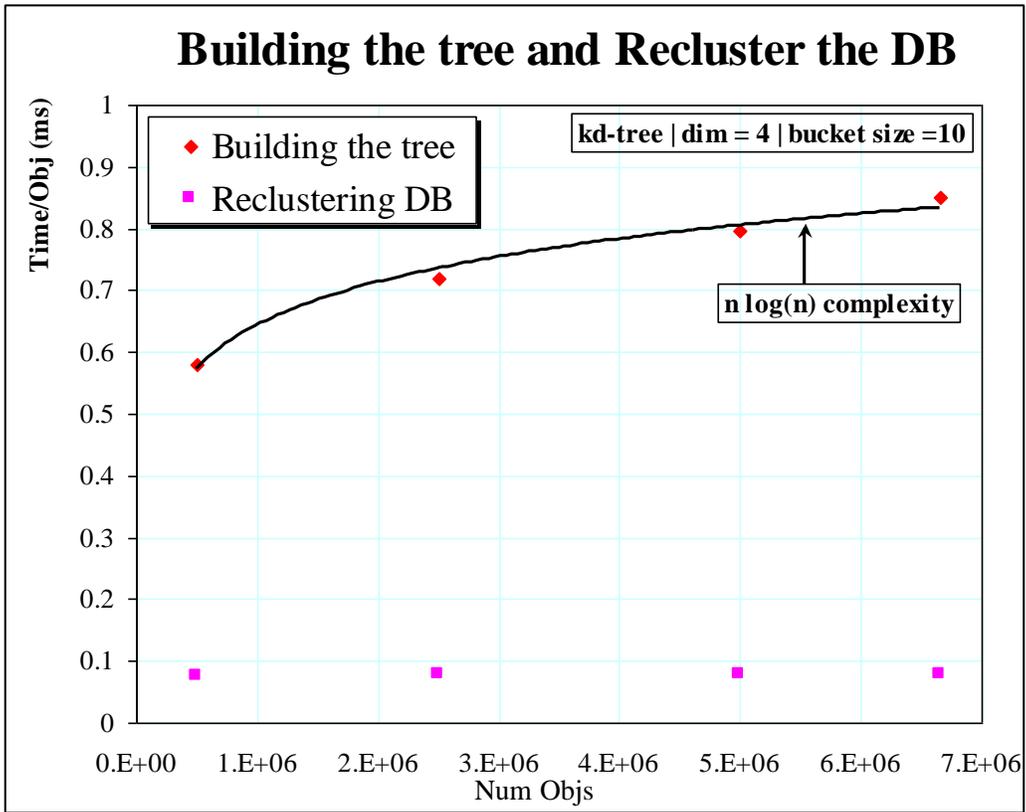


Figure 1

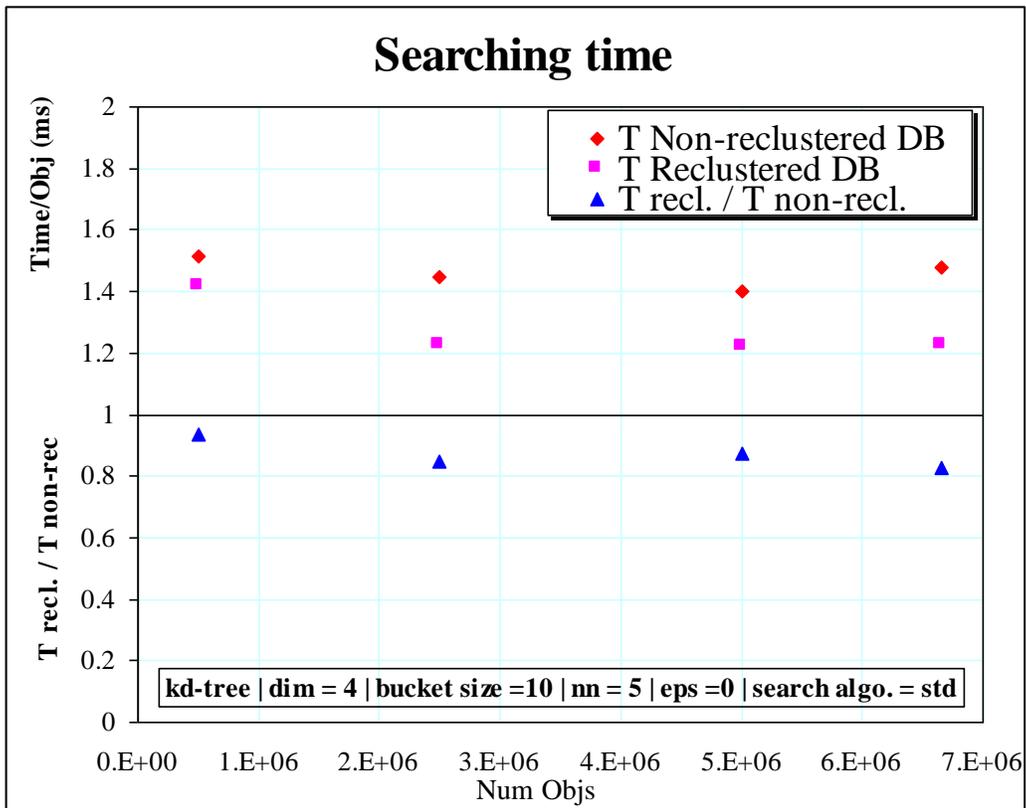


Figure 2

References

¹ James Annis, Gabriele Garzoglio, Kurt Ruthsmandorfer, Chris Stoughton, 'The Terabyte Analysis Machine Project, the Distance Machine', <http://projects.fnal.gov/act/tam/TAM.htm>

²The SDSS Science Archive www.sdss.jhu.edu/ScienceArchive/home.html

³ J.L. Bentley. 'K-d trees for semidynamic point sets'. In *Proc. 6th Ann. ACM Sympos. Somput. Geom.*, pages 187-197, 1990

⁴ Gabriele Garzoglio, 'Preliminary ANN/Objectivity Tests', http://projects.fnal.gov/act/tam/Prel_Perf_tests.htm