

The Terabyte Analysis Machine Project

The Distance Machine

James Annis, Gabriele Garzoglio, Kurt Ruthsmandorfer, Chris Stoughton

Abstract: The Terabyte Analysis Machine is a cluster designed to allow research into the use of large scientific databases. Our initial program is to explore efficient database re-indexing and re-partitioning as a way to take maximal advantage of sophisticated algorithms for use in specific astronomical problems. The model is to allow the scientist to redistribute the database to make very efficient a specific problem, e.g., finding the k th nearest neighbor, and allow them to develop on that database for a month before moving onto the next redistribution scheme. We will discuss the main ideas of a prototype design, giving a few details on the k th nearest neighbor repartitioning module.

Introduction

The databases of modern physics and astronomy contain hundreds of millions of objects and have total sizes of order a Terabyte. Spatial data structures provide efficient multi-dimensional access methods to these large datasets. The database designers make use of spatial data structures that provide good general purpose access to the data. It is clear, however, that the best data structure to use depends on the problem to be solved. That is, the optimal indexing and partitioning of the data is most often algorithm dependent.

The distance machine is an attempt to build an infrastructure to allowing physicists and astronomers access to the powerful algorithms requiring repartitioning and re-indexing most relevant to their particular problem, without dealing with the low level issues of remote database access, authentication and parallelization.

Proposal

We propose to prototype a facility that acquires, re-indexes, and re-partitions a

database. We aim to make it straightforward for the physical scientist to incorporate a spatial data structure based algorithm into the facility to guide the re-indexing and re-partitioning. We choose as our example algorithm k th nearest neighbor finding. This problem is an area of active research in the computer science community (there are no exact solutions for dimensions greater than 10 that are faster than a brute force scan), plays a central role in SPH numerical calculations of structure formation and, we believe, will form the basis of a state-of-the-art galaxy cluster finding algorithm. Brute force solutions to the k^{th} nearest neighbor problem work, given enough compute power and time, but for dimensions of order 5, there are significant speed-ups that can be obtained from tree-based algorithms. We wish to explore the usefulness of a facility where a database is repartitioned for an intermediate timescale, on order of a month, allowing the astronomer an efficient code development, testing, validation cycle with the data.

Scales

The SDSS database will have:

- 500 Gbytes database
- 250 million objects
- 100 million galaxies
- 2.5 Kbytes object size
- 500 byte tag object size (example of tag with about 50 parameters of interest for the specific analysis)
- 50 Gbytes tag database

Moving the tag database:

- 50 Gbytes at 10 Mbytes/sec => 1.5 hours to transfer over network
- 50 Gbytes at 30 Mbytes/sec => 0.5 hours to write to disk

Dimensionality of the problem:

- 120 parameters per object
- 5 parameters used for cluster finding: ra, dec, g-r, r-i, and i'

Prototype

The Distance Machine does the following steps:

1. extracts a tag database from SX
2. places the tag database on local machines
3. re-indexes and re-partitions the tag database
4. queries the tag database and sends results to problem specific analysis tools

The SDSS database architecture developed at JHU is called SX¹; it currently relies on Objectivity. The distance machine will work on a subset of the SX, in a local tag database.

A couple of techniques have been considered to implement the tag extractor module. The easiest one uses the analysis engine mode of SX. This technique, already implemented, queries SX for a subset of parameters from the objects of interest. The tag objects are sent from SX to TAM via a socket and then stored on local disk. The

analysis engine approach has the advantage of being independent of underlying database.

We are, however, planning to use Objectivity for the local database; this opens up some interesting possibilities. A tool of general interest could be built using a tag extraction layer intermediate between SX and Objectivity making use of Globus² tools, similar in spirit to the CERN/Caltech CMS re-clustering tools³. This tool could extract the tag objects from Objectivity, store them remotely as Objectivity databases, and ship them to the extractor module via globusFTP agents. As such code exists, and naturally links to a grid infrastructure, we will examine this approach closely.

Both approaches need the specification of a tag schema. The tag extractor needs the knowledge on the structure of the data to query SX for the subset of parameters of interest and for saving the objects locally. The new tag database will then rely on this schema for all the re-indexing, re-partitioning and querying operations. Having chosen Objectivity as tag database, DDL (Data Description Language) seems the natural choice to produce easily the code that has to be linked to the TAM module implementations. However, we are still investigating other options: there is soon to be technology inside SX to make defining new schema easy. This could allow the distance machine to offer facilities to deal with complex data structure keeping the coding of the schema a relatively easy process.

The same schema definition will be part of the repartitioning module also. The core of our kth nearest neighbor module makes use of the Approximate Nearest Neighbors library (ANN)⁴. This library is well commented and implements several of the partitioning and searching algorithms available in 1998; also very important, it's "off the shelf". The most recent version of this C++ library (v0.2) represents a vector in the space as a pointer to an array of coordinate. The library offers

facility to build either a k-d tree or a b-d tree of the data and to search the k^{th} nearest neighbor within this structure. This approach is fast and immediately of use for a data set size that fits into memory. As mentioned, we operate with tag databases of the order of 50 Gbytes; therefore the modification of this library to deal with data laid on disk is necessary. The choice of Objectivity offers an easy way of implementing the persistency of the tags. We are implementing a general-purpose wrapper that stores into memory a reference to the persistent object (plus other information of internal use, like the number of copies in memory of the same object). Overloading the operators of assignment and indexing, this wrapper presents ANN transparently a static representation of the space (as big as we need) in about ten bytes per object.

At this point, the structure of the k-d tree, that has an object-oriented representation inside ANN, can be stored using again Objectivity. The leaf nodes of the tree are the natural re-indexing of the database for efficient nearest neighbor queries. That is, the natural re-partitioning of the database stores each non-empty node, whose size may range from 8 to 32 Kbytes (bucket size), sequentially in the database. Reasonable values of the Objectivity parameters are 32 Kbytes for the page size, 6 - 60 Mbytes for the client cache size, 1 - 10 Mbytes for each container. This fine-tuning, aimed to enhance performance, has yet to be investigated.

We believe this approach is generalizable to other algorithms, not necessarily k^{th} nearest neighbor ones, allowing them to be placed in the infrastructure with little effort. Clearly a long term aim is to allow the definition of schema and interfaces (i.e. to the wrapper) that make possible the re-indexed and re-clustered/re-partitioned of a database on the fly.

Lastly, and perhaps outside the range of the prototype, is the challenge of parallelizing

this operation. Divide and conquer techniques work well for this work, and the TAM is designed to make divide and conquer techniques easy. TAM is a cluster of (currently) 5 Linux machines, with 70Gbytes of local disk each plus a fibre channel connection to a common terabyte RAID box. The information of the disk layout is needed at the level of the re-clustering module, the design of which must include the notion of parallelization of the query process. There is more than one algorithm that can be implemented in order to exploit the parallel nearest neighbor search. We feel most confident with the so-called shadow technology, which defines replicated border regions that are included in the search of those query points that lay inside the non-replicated portions.

Schedule

1. Tag extractor module: Jan 31
2. Integrating ANN with Objectivity: Feb 14
3. Re-clustering module: Feb 28
4. Saving/Retrieving ANN trees: Mar 21
5. Set up an analysis engine: Mar 31

References

- ¹ The SDSS Science Archive
www.sdss.jhu.edu/ScienceArchive/home.html
- ² The Globus Project, www.globus.org
- ³ Koen Holtman, "Data Clustering Research in CMS", Proceeding of CHEPP 2000, Padova, Italy,
kholtman.home.cern.ch/kholtman/
- ⁴ ANN Programming Manual; David M. Mount, Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland,
www.cs.umd.edu/~mount/ANN/