Column-Oriented Table Access Using STIL: fast analysis of very large tables

Authors:
Mark Taylor, Bristol University, UK
Clive Page, Leicester University, UK
Work supported by:
VOTech, PPARC, Astrogrid, Starlink

Abstract

By use of column-oriented storage and file mapping, great improvements in efficiency over more conventional methods can be made for some important kinds of access to large and very large tabular datasets. These techniques have been implemented in the STIL library, enabling their use in TOPCAT (an interactive graphical tool for table analysis) and STILTS (a command-line table analysis suite). Benchmarks are presented which show certain common analysis tasks running 10–40 times faster than their MySQL equivalents. Applied to datasets in the range hundreds of Mbyte to hundreds of Gbyte this speedup can be put to good use both on the desktop and at the data centre to bring new regimes of data exploration within practical reach.

Introduction

When dealing with tabular data stored on disk, the most straightforward way to operate is to read the entire file into memory and then to do the processing. This approach works well for small datasets, but for large tables, especially when the required memory begins to exceed available physical memory, it can become inefficient or unworkable, and some kind of on-demand access to the data on disk becomes necessary.

There is no shortage of datasets for which this is a genuine issue. On the astronomer's desktop it is true that one trend in Virtual Observatory-type working is to retrieve small (i.e. small enough to fit in memory) subsets of large server-based datasets for local analysis. However the opposite approach in which a whole dataset, or a substantial part of one, is directly available to the astronomer can sometimes be much more effective, especially for exploratory analysis or revealing unexpected relationships. At the data centre the size of survey catalogues continues to grow, and while relational databases are widely seen as the only option for handling these, there are some common query types for which RDBMS performance is poor.

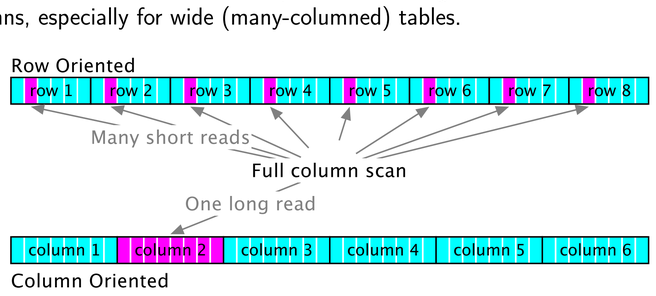
For such on-demand access, whether sequential or random, the way the data is arranged on disk and the way in which it is accessed can have a major impact on performance. This paper describes approaches to these issues which provide highly efficient data access, and presents library and related application software which makes use of it to provide practical benefit for client- and server-side data access to large scale data.

Techniques

The central problem of providing efficient disk I/O is to ensure that disk accesses and system I/O calls are infrequent. It is *much* better to do one read of 1000 values than 1000 reads of one value. The techniques explained here have been employed to help attain this goal.

Column-Oriented Storage

There are two obvious arrangements for storing table data on disk: *row-oriented* and *column-oriented*. Most common table storage formats (FITS, VOTable, CSV, nearly all RDBMS) are basically row-oriented. This is good for reading all the columns from a few rows, but poor for scanning the whole of one or a few columns, especially for wide (many-columned) tables.



Many common analysis operations benefit from column-oriented access, e.g.:

- Scatter plot/density map/histogram plotting
- Row selection based on an unindexed column or a combination of columns
- Univariate or multivariate statistic calculations

File Mapping

Some operations require random access on a table, which means reading a few bytes here and a few bytes there. Naïve implementations would result in very poor performance. In principle it is possible to improve matters by reading and caching larger blocks near each site, but it is difficult to decide on optimal caching strategies (what size blocks to read, how long to keep them).

By using *file mappping* (Unix `mmap(2)` system call or Java `FileChannel.map` method) rather than buffered or unbuffered seek/read operations you can get the Operating System to take care of this for you. Highly optimised OS routines for block caching are then used automatically which usually results in good performance for a wide range of access patterns. As a bonus, file-mapped reads are typically faster ($\times \sim 2$?) than normal reads. There are one or two OS-dependent issues with this technique, but tests have found it working well. One issue to note is the limitation of file size by available address space. In practice this means that for multi-Gbyte datasets, a 64-bit OS is required.

Implementation

The Starlink Tables Infrastructure Library (STIL) is a general purpose multi-format library for I/O and processing of astronomical tables. Its pluggable architecture makes it suitable for testing out the ideas described here. STIL table handlers were implemented which provide mapped access to column-oriented data, and these could be used without changes to application code.

A new column-oriented file format, dubbed "colfits" was introduced for this purpose. It is a variant of FITS — a table is represented by a one-row (`NAXIS2=1`) BINTABLE extension, in which each cell of the single row is a vector containing all the values in the column. The resulting file is perfectly legal FITS, though general purpose FITS handling software may or may not make much sense of it.

It was easy using the STILTS `tcopy` command to convert between these storage formats, including exchanging data with a MySQL database.

Benchmarks

Some full column queries were performed on datasets in various forms to assess performance. Two datasets were used:

XSC: 2MASS Extended Source Catalogue (1,647,599 rows $\times$ 391 cols $\approx$ 2.2 Gb)

PSC: 2MASS Point Source Catalogue (470,992,970 rows $\times$ 61 cols $\approx$ 111 Gb)

Two types of query were run on each dataset:

STAT1: calculation of mean, variance etc on a single column

SEL2: row selection based on difference of two columns

and three data storage systems were tested:

MySQL: MySQL 4.1.20 using unindexed MyISAM tables

colfits: STILTS using column-oriented file-mapped FITS

fits: STILTS using row-oriented file-mapped FITS

Some representative results are as follows:

Data	Test	MySQL	colfits	fits
XSC	STAT1	65	2.0	51
XSC	SEL2	66	4.7	89
PSC	STAT1	3390	105	2321
PSC	SEL2	3422	397	2417

(Timings in seconds)

The STILTS/colfits results are 10–40 times faster than the MySQL ones for these queries. Reducing XSC queries to a few seconds and PSC queries to a few minutes qualitatively changes the kinds of work that an astronomer can do with these datasets, bringing interactive modes of data investigation within reach.

Comparison with RDBMS

In most cases very large astronomical tables are stored in relational database management systems (RDBMS). These are general purpose packages highly optimised for large datasets and for many purposes they perform very well. Some column-based queries can be handled very fast, where they can take advantage of precomputed indexes. But unindexed queries are typically slow[a].

RDBMS have some other disadvantages for use in data analysis. In some contexts their complexity is a barrier to use: the administrative overhead of installation and configuration is appropriate for data centres, but less so for personal or read-only access on local disk. Integration with application code, based as it is on SQL, tends to be somewhat unwieldy making interactive applications hard to build. Metadata handling is also poor.

Much of the sophistication of RDBMS, for instance data updates and access controls, is not required for exploratory analysis of astronomical catalogues. STIL provides a simpler alternative for storage of tabular datasets which is truly scalable, faster for some tasks, and amenable for use in application packages. It in no way however claims to supplant use of RDBMS altogether for catalogues, lacking for instance persistent precomputed indexes.

[a] At least one RDBMS, *Sybase-IQ*, uses column-oriented organisation of data on disk. However, the licence for this is currently very expensive, which makes it unsuitable for most astronomical applications.

Software

The software discussed here is written in Java (hence highly portable and easy to install), fully documented, well supported, and available under the GNU Public Licence. For more modest sized tables it works equally well with other formats including normal FITS and VOTable.


STIL: Starlink Tables Infrastructure Library
http://www.starlink.ac.uk/stil/

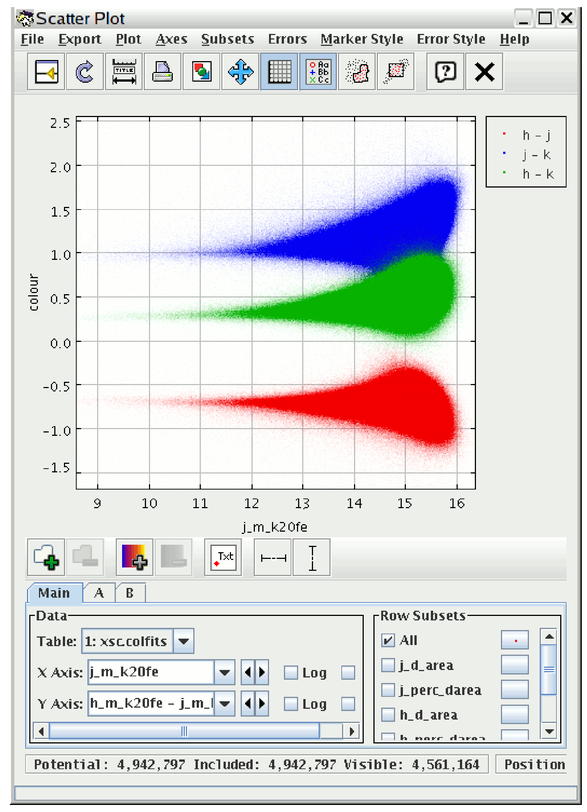

STILTS: STIL Tool Set
http://www.starlink.ac.uk/stilts/


TOPCAT: Tool for OPerations on Catalogues And Tables
http://www.starlink.ac.uk/topcat/

Applications

Desktop TOPCAT

TOPCAT is a powerful and easy to use interactive graphical tool for astronomical table analysis. It is based on STIL. Many of the features it provides, such as various kinds of plots, crossmatching, row selections etc necessarily require full-column scans of tables. Using the techniques described here, large datasets (up to about 10^7 rows and any number of columns) can be investigated interactively — plots, row selections etc take only a matter of seconds.

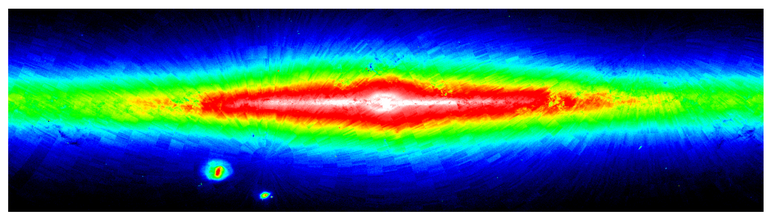


TOPCAT screenshot showing a three-way colour-magnitude plot from a colfits-format copy of the entire 2MASS extended source catalogue (1.6 Mrow $\times$ 400 col $\approx$ 2 Gb). 4.5 million points are visible. Plots like this can be drawn, zoomed, modified, exported etc in a very few seconds on platforms with modest resources.

Desktop STILTS

STILTS is a command-line table analysis package based on STIL. It offers many of the same features as TOPCAT (crossmatching, selection and general table manipulation), and a number of others, but using a scriptable and non-graphical interface. Unlike TOPCAT, many operations run in streaming mode, so that there is no limit to the size of tables which can be processed.

Converting very large tables from other forms (e.g. RDBMS, normal FITS, CVS, VOTable, ...) into colfits format can itself be done using the STILTS `tcopy` command. Once this is done, STILTS analysis operations can easily be run in column-oriented mode.



FITS file representing a density map (2-d histogram) of all point sources in the 2MASS Point Source Catalogue (470 Mrow $\times$ 60 col $\approx$ 111 Gb) in the galactic plane. It was generated using the STILTS `tcube` command in about 10 minutes. Note the striped features corresponding to survey frame-edge artifacts, not obvious except from a plot like this. This kind of graphic would be both slow and difficult to obtain using other software.

Data centre STIL/TS

Datasets of the size of the 2MASS PSC are not often found on the desktop, but are common at data centres, usually stored in relational databases. Some client requests for this data will be of the kind which can be serviced efficiently using usual RDBMS facilities (SQL SELECTs), but others, requiring full column scans, will not. Very often this latter kind, even if small in number, can dominate the I/O and computational load on the server. A server could keep two replicated copies of the data, one in RDMBS format and the other in a column-oriented format such as colfits, using the more efficient form for each incoming request as appropriate. This arrangement is not currently in place at any data centre (as far as the author knows), but some interest has been shown.

Your code here...

Feel free to come up with your own scenarios for usage of the STIL library or STILTS application suite.