Astronomical Data Analysis Software and Systems XVII **P4.6**
ASP Conference Series, Vol. 394, © 2008
R. W. Argyle, P. S. Bunclark, and J. R. Lewis, eds.

Column-Oriented Table Access Using STIL: Fast Analysis of Very Large Tables

Mark B. Taylor

Department of Physics, University of Bristol, UK

Clive G. Page

Department of Physics and Astronomy, University of Leicester, UK

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 the established table analysis applications TOPCAT and STILTS. 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 center to bring new regimes of data exploration within practical reach.

1. 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 such as a survey catalog, 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 center the size of survey catalogs 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 are arranged on disk and the way in which they are 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 access to large scale data.

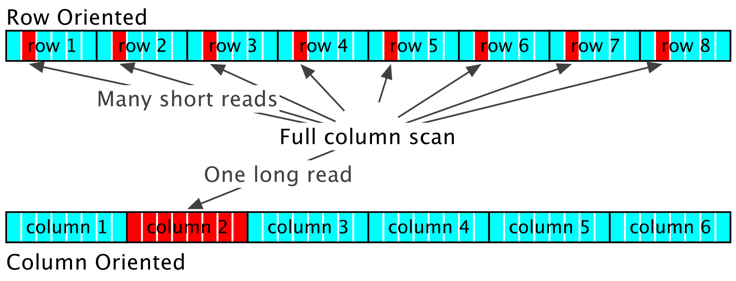

Figure 1. Schematic of a full column scan using row-oriented and column-oriented storage.

2. Techniques

The central problem of providing efficient disk I/O is to ensure that disk access and system I/O calls are infrequent. This section discusses two of the data access techniques used in the current study to achieve these goals. Neither one of these is particularly novel in general or in astronomical contexts; column-oriented table handling in the MIDAS and STSDAS packages and file mapping in the Starlink HDS library provide examples of prior art of a couple of decades' standing. However use of these techniques is far from ubiquitous and their importance in supporting the reported performance at high data volumes makes it worthwhile to review them here.

2.1. 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, as illustrated in Figure 1, poor for scanning the whole of one or a few columns. This is especially true for wide tables, increasingly the norm for astronomical survey data; for instance 2XMM has ~ 300 columns and the SDSS PhotoObj table has ~ 500.

Many common analysis operations benefit from column-oriented access, for instance full-table data visualisation, row selection based on an unindexed column or a combination of columns, and univariate or multivariate statistical calculations.

2.2. File Mapping

Some operations, for instance sorting, crossmatching, and use of temporary or persistent indices, require random access on a table, which means reading a few bytes here and a few bytes there. Naïve read-on-demand 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 mapping* (the 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 somewhat faster 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 mapped file size by available address space. In practice this means that for multi-Gbyte datasets, a 64-bit OS is required.

2.3. Implementation

The Starlink Tables Infrastructure Library (STIL[1]) is a general purpose multi-format library for I/O and processing of astronomical tables. By design the library itself makes few assumptions about data access methods or storage layout, but has a pluggable architecture which permits experimentation with different data access layers. This makes it highly suitable for testing out the ideas described here. STIL table handlers were implemented which provide mapped access to column-oriented data, enabling use of these techniques in established applications without requiring changes to the 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 it may or may not be intelligible to general purpose FITS handling software.

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

3. Benchmarks

Some full column queries were performed on datasets in various forms to assess performance. The following datasets were used:
XSC: 2MASS Extended Source Catalog (1,647,599 rows $\times$ 391 cols $\approx$ 2.2 Gb)
PSC: 2MASS Point Source Catalog (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 between two columns
and three data storage systems were tested:
MySQL: MySQL 4.1.20 RDBMS using unindexed MyISAM tables
colfits: STILTS using column-oriented file-mapped FITS
fits: STILTS using row-oriented file-mapped FITS

Table 1 shows some representative results. For these and similar queries STILTS/colfits run times are 10–40 times faster than the MySQL ones. In cases where columns can be cached in memory, not shown here, subsequent queries can be faster still.

[1] `http://www.starlink.ac.uk/stil/`

Table 1. Timings in seconds for column scan benchmarks.

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

4. Discussion

The dramatic query time reductions exemplified in the previous section can qualitatively change the kinds of work that an astronomer can practically do on large datasets, bringing interactive modes of data investigation within reach. The two datasets used there (2MASS XSC and PSC) represent two interesting, and illustrative, regimes of data size.

The XSC, at 1.6 Mrow and 2 Gbyte, is of a size which would conventionally require some pre-selection prior to interactive analysis; for instance an astronomer might query a copy of the database using some VO or data center-specific protocol to retrieve a small selection of rows and columns and then perform plots or other analyses on that selection only. However, using the software presented here it is quite feasible to acquire the entire dataset (2 Gbyte is no longer too large to download or to store locally) and work interactively on it using TOPCAT[2], which is a graphical tabular analysis application based on STIL. For instance a colour-magnitude plot of all objects in the XSC can be generated in just a few seconds even on a machine of modest specification.

The PSC, at $>$ 100 Gbyte, is far too large to download to order. But data centers which need to service queries entailing full-column scans could make use of data in column-oriented form alongside their existing representation in a conventional database. Duplicating the data in this way, while increasing disk space requirements (scarcely an economic constraint these days) could substantially reduce I/O and CPU load on servers which are dominated by full-column scan-type queries. Since such queries are much the most expensive operations for conventional databases, such reductions are feasible even where these queries form only a small proportion of those received. This approach could be implemented using either custom application code based on STIL, or using STILTS[3], a suite of command-line tools based on STIL.

Acknowledgements. This work has been funded by the VOTech and AstroGrid projects, the former Starlink project and the former Particle Physics and Astronomy Research Council.

[2]`http://www.starlink.ac.uk/topcat/`

[3]`http://www.starlink.ac.uk/stilts/`