Paper 115-25

Efficiency Techniques for Accessing Large Data Files

By Andrew Wilcox
Amadeus Software Ltd 1999 ©

Abstract

SAS® software offers the programmer a multitude of ways to enable fast data extraction from large SAS datamarts. The techniques discussed include the use of base SAS software i.e. indexes, Screen Control Language, Structured Query Language, HOLAP techniques for client server architecture and Scalable Performance Data Server component.

This paper discusses some of the extraction options available across the broad spectrum of SAS versions and operating system platforms to minimise overall processing times.

The areas to consider when talking about efficiency techniques include, CPU time, I/O, system memory usage, programming time and disk space.

Introduction

The techniques discussed in this paper are:

Using Keep, Drop and Where options.
Data restructuring.
Remote Processing.
Compressing SAS data sets
Using Index's.
Using SAS Views.
Using SQL.
Source Control language (Source Component Language).
Using MDDB's.
Using HOLAP.

Emerging Technologies

Scalable Performance Data Server component.
Efficiency Enhancements for Version 7/8 of the SAS System.

General Efficiency Techniques

The most important efficiency factor in any Data Mart is to familiarise yourself with the data. Once the current information structure is understood we can explore the efficiency techniques offered by the SAS System.

1. Using Keep, Drop and Where

The most common form of efficiency methods in base SAS code is to use **DROP, KEEP** and **WHERE** options on the SAS data sets/views the system references.

Simple example

```
data report;
   set sasuser.class;
   If age > 30 then output report;
run;

proc print data=report;
   var age gender;
run;
```

This code can be replaced with

```
proc print   data=sasuser.class (keep=age
gender where=(age > 30));
run;
```

By applying DROP or KEEP statements to all procedures that reference data sets/view overall performance savings of 10-20% CPU time are not uncommon.

2. Data Restructuring

Often the most important yet easily forgotten form of system tuning is the transposition of data from a 'vertical' structure to a 'horizontal' layout to help Where clause performance.

Consider the following vertical list storage file.

Data Set One

WEEK_NO	DATE	CATEGORY	CASES
1	20/06/1999	A	10
1	20/06/1999	B	12
1	20/06/1999	C	15
2	27/06/1999	A	12
2	27/06/1999	B	17
2	27/06/1999	C	11

This could be transposed into data set Two

A	B	C	WEEK_NO	DATE
10	12	15	1	20/06/1999
12	17	11	2	27/06/1999

Now any data lookup technique only has to process two observations rather than six.

3. Remote Processing

SAS/CONNECT Software enables users who use a client install of the SAS System to connect to a remote server. Thereby taking full advantage of the servers greater storage capacity and processing power. Once a signon script has been executed uses may communicate with the server by using RSUBMIT blocks, files may be moved from client to sever and vice versa using either PROC UPLOAD/DOWNLOAD or RLS (Remote Library Services).

For example.

```
rsubmit;

proc upload in=sasuser out=work;
   select class;

data sample;
   set class(where=(age > 30));

proc download in=work out=sasuser;
   select sample;
run;

endrsubmit;
```

4. Compressing SAS Data Sets

Compressing a SAS data set can improve data access times on certain table structures. This is due to the reduced I/O required to read the reduced data file size.

Example

```
data sasuser.huge(compress=yes);
   set sasuser.huge;
run;
```

Space savings of 20-30% are not uncommon, however this technique proves most effective on data sets that contain mainly character data.

However extra CPU time may be required to decompress the information during an extraction request.

5. Using Indexes

Indexing is generally accepted as the industry standard in providing users with performance benefits in dealing with large data files, just as a books index helps locate relevant information quickly.

However, introducing index's into a data warehouse can be fraught with difficulty.

The primary concern to be addressed is SPACE…. Not outer space but disk space. Currently mainframe disk packs cost about £2000 for 2.4 Gb worth of space and there will be information bottlenecks and motherboard considerations as to whether the server can be upgraded, on an NT server 20 Gb will cost about £400 so space is not much of an issue.

Creating an index will often increase the required disk space by 20-50 % depending on the operating system. If adding extra packs is not a viable option then try using the COMPRESS facility the SAS System offers as a DATA step option.

Once the space issue has been addressed, deciding on your indexing method is the next hurdle, here the problem is which method to use to create an index, four methods below are discussed.

PROC datasets

Example

```
proc datasets lib=sasuser mt=data;
   modify class;
   index create composit=(status);
run;quit;
```

PROC SQL

```
proc sql;
   create index status on
   sasuser.class(status);
quit;
```

DATA Step

```
data sasuser.class(index=(status));
   set sasuser.class;
run;
```

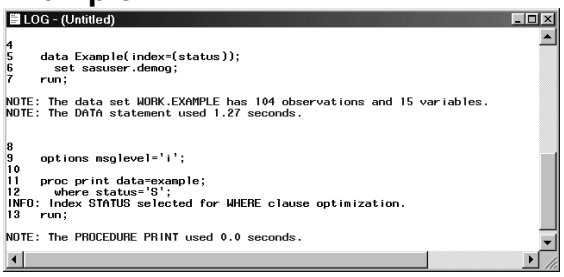
SCL

```
dsid=open('sasuser.class','V');
rc=icreate(dsid,'status', ' ','nomiss');
rc=close(dsid);
```

Each method has its own merits and all must be tested to help reduce over CPU processing times, one immediate improvement in the creation of index's is to make sure the data is pre-sorted by the indexing variables.

Once an index has been defined any where clause syntax applied to the database will take advantage of the index, if the SAS System is using the index a message will appear in the log so long as the option msglevel='I' is active.

Example



6. Using SAS Views

Through the use of SAS views a general efficiency is made in terms of disk space since the view contain a link back to the original source data as opposed to containing the data themselves.

With today's large data volumes SAS Views are often the interface between the 'front-end' SAS reporting system and the 'back-end' database (for example, DB2, ORACLE, Ingress etc).

Accessing information held within a view will not be as fast as reading information direct from a native SAS data set, as mentioned previously a common efficiency technique would be build an index onto the SAS file, but with a view this is impossible as it contains no data.

One common technique of accessing information quickly from a view is to 'slice and dice' the view into component SAS data sets, these resulting data sets can be indexed. Unfortunately this is not always possible due to available disk space requirements, as we are in all intents and purposes, copying the main data mart.

7. Using SQL

The SQL (Structured Query Language) programming language was introduced into the SAS System in version 6.06.

The main advantage of using SQL to access information held in a view/data set (or a sequence of views/data sets) lies within its internal sort capability and its 'SET-WISE' read method.

A traditional MERGE statement would require a PROC SORT step which would output the information to a SAS data set before an extract could be requested via a MERGE. Using SQL joins instead eradicates the need to output any information to a SAS data set before the query has been requested.

The internal process used by the SQL programming language can be tracked using the _METHOD option.

For example

```
proc sql _method;
   create table sample as
   select * from table1 a full join
   table2 b
   on a.key=b.key;
quit;
```

Performing table lookups is a primary use for this programming language however, SQL2 can only perform a maximum of 16 table joins in one SQL step compared with 1000's in the SAS DATA step.

Performance issues with SQL can also become an issue if a query requests more than 10% of the whole database, this is due to the 'SET-WISE' read method SQL uses for its data lookups as this read method is very memory intensive.

8. Screen Control Language. Source Component Language (Nashville Release)

The SCL language offers the programmer a multitude of functions capable of performing data extraction. Many of these SCL functions are now available in the SAS data step however, there are some issues to be aware of.

The use of the ICREATE function to create an index and subsequently a data step where clause can prove an explosive combination especially on something like a UNIX server.

For Example SCL sort & index code

Perform an SCL sort

```
dsid=open('sasuser.class','U');
rc=sort(dsid,'gender status');
rc=close(dsid);
```

Create an index

```
dsid=open('sasuser.class','V');
rc=icreate(dsid,'status',' ','nomiss');
rc=close(dsid);
```

However, using SCL functions such as LVARLEVEL and SEARCHC can prove to be slow on large SAS data sets as the observations are held in active memory until a match is found.

10. MDDB's

An MDDB is a specialized data storage facility that stores summarized data for fast and easy access. Users can quickly view large amounts of data at any cross-section of the dimensions.

A dimension can be any vision of the data that makes sense, such as time, geography, or product. Users create and update multidimensional databases using SAS/EIS software or PROC MDDB. Once an MDDB is created, it can be copied or transported to any platform which supports the SAS System.

Example

```
proc mddb data=sasuser.class
   out=work.mddb;
   class gender status grade dept;
   var salary / sum;
   var age / sum;
   hierarchy gender status /name=across
   display=yes;
```

```
   hierarchy grade dept / name=down
   display=yes;
run;
```

This new MDDB file will contain all the crossings of the four classification variables. Users are able to access this information via some SAS/EIS objects and through SCL or by using the SASSFIO engine included as part of SAS/CONNECT Software.

10. Hybrid On-Line Analytical Processing (HOLAP)

SAS/EIS® Software provides dynamic drill capabilities to users without extensive SCL programming with the table editor. Some EIS objects, such as the multi-dimensional report viewer, have the capability to support client server queries. To enable this SAS/MDDB® Server and HOLAP are required. (HOLAP is available free from SAS institute or as standard in SAS 6.12 release TSO55 and above).

summary files remotely, thusHOLAP allows an MDDB to be split into its individual summary components which live on a server whilst a 'dummy' MDDB lives on the client. Once HOLAP has been setup, any processing of the 'dummy' MDDB involves querying the providing lightning fast response times in a front end SAS application.

The draw back is making sure you have enough disk space for the summary files and in scheduling any updates. Updates are ok if they are weekly, but more frequent updates might prove problematic.

Emerging Technologies

Scalable Performance Data Server component

Scalable Performance Data Server software is a fully integrated and seamless way to access large volumes of data. It serves large numbers of concurrent users by utilizing the latest parallel processing and data server capabilities.

Preliminary tests show that Scalable Performance Data Server organizes and exploits your data much faster than is currently possible with parallel processing options offered by other database vendors -- 15 percent to 20 percent faster in most cases.

Scalable Performance Data Server software is currently available on the following platforms for Version 6 Release 6.12 of the SAS System:

Release 2.1 - Sun, HP, NT, IBM(AIX)

(Source SAS Institute 1999)

Efficiency Enhancements for Version 7/8 of the SAS System

With the rollout of version 7 of the SAS System extra indexing features have been implemented into base SAS to help cope with efficiency issues that arise in large ever expanding data marts.

Traditionally a SAS data mart is created from a data base system via a weekly batch job, the new information is then appended into the existing data mart. This process works well until a table size grows large enough to present performance problems. These problems usually surface when composite index's have to be rebuilt on the 'whole' file.

Version 7 of the SAS System is introducing an indexing feature that will alleviate some of this problem. In version 7, as long as the base and appended data sets have the exact same variables, an internal sort will be preformed on the values required to complete the append process for each index in the base data set. This improvement is added along with some other internal short cuts (more efficient appends for observations that are added with the same value of an indexed variable) , and showed good results in internal SAS benchmark test.

Version 8 of the SAS System will introduce a parameter to the index delete request that will allow the use of an option _ALL_ that will delete all indexes without an internal cleanup step between each index delete being executed. This will help speed up the index deletes portion of this alternative, but the primary expense is in rebuilding the indexes.
(Source SAS Institute Cary)

Summary

The techniques discussed in this paper are to provide the programmer with data extraction options. No one method is better than another all have individual merits depending on the host data.

References

SAS Language and Procedures: Usage, Version 6, First Edition, 1989
SAS Institute, Inc., Cary, NC, USA

Client/Server Computing with the SAS System: Tips and Techniques, 1995.
SAS Institute, Inc., Cary, NC, USA

SAS Guide to the SQL procedure: Usage and Reference, Version 6, First Edition, 1989.
SAS Institute, Inc., Cary, NC, USA

SAS/EIS® Technical Report: HOLAP Extension, Release 6.12
SAS Institute, Inc., Cary, NC, USA

SUGI 22,23,24 Papers

Large Scale Data Warehousing with the SAS® System.
Tony Brown, SAS Institute Inc., Dallas, TX
Leigh Ihnen, SAS Institute Inc., Cary, NC
Jim Craig, SAS Institute Inc., Austin,TX

Unix Large File Processing Secrets
Karsten Self, PM Squared Inc., San Francisco, CA

Learning About Your Data: Tips and Techniques for Looking at Large Files
Sandra D. Schlotzhauer, Schlotzhauser Consulting
Bob Anschuetz, Quintiles, Inc.

Faster SAS® Jobs and Fewer Passes Via Data Step Views
Steven First, Systems Seminar Consultants., Madison, WI

Efficiency Ideas for Large Files
J. Meimei Ma, Quintiles, Research Triangle Park, NC
Andrew H. Karp, Sierra Information Services, Inc., San Francisco, CA

Contact Information

Amadeus Software Ltd
13 Corn Street
Witney
OX8 7DB
England

Telephone	+44 (0) 1993 775454
Fax	+44 (0) 1993 700577
E-mail	info@amadeus.co.uk
Web Page	www.amadeus.co.uk

Copyright Notice

Trademark Notice