ABSTRACT

This paper compares and contrasts the external performance and storage management characteristics of the SAS Versions 5 and 6 data library implementations under MVS. Performance comparisons of data set loading, data set reading, and directory management functions are presented. Also included are comparisons of library space requirements under several scenarios and some discussion of tuning considerations specific to Version 6 libraries.

INTRODUCTION

Under most operating systems, the SAS data library is implemented as an aggregate of files in the native file system of each operating system. Under MVS, SAS data libraries have historically been implemented using a "bound data library" technique. That is, multiple SAS data sets are stored in a single MVS file. Conceptually, this is analogous to the MVS Partitioned Data Set (PDS) file format. In fact, the SAS data library was originally implemented as a PDS. Beginning with SAS 76, the library format which is currently used in SAS 5.18 was first delivered and has remained substantially unchanged since that time. The data library implementation in Version 6 is the first significant departure from the SAS 76 implementation. Three papers being presented at SUGI 14 give different perspectives on the data management features of SAS Version 6. See (Clifford 1989) for a discussion of data management features and options available on all hosts supported by release 6.06. See (Bowman 1989) for details on the implementation of the SAS Data Library for Version 6 under MVS. This paper discusses external characteristics and performance properties of SAS data libraries under MVS.

SAS DATA LIBRARY STORAGE MANAGEMENT CHARACTERISTICS

Version 5 Space Management Characteristics

Version 5 data libraries are implemented as multiple members stored in a DA-type disk file. Each member starts on a track boundary and consumes an integral number of tracks in the library. Therefore, the smallest unit of allocation has grown with successive generations of IBM disk from roughly 7k on the 2314 to 13k on the 3330, 19k on the 3350 and finally 46k on the 3380. This trend has had an adverse effect on the space requirements for data libraries which contained a large number of small members. The following example illustrates the type of wasted space problem that can occur in a Version 5 SAS data library.

Simplified Example of Member Layout in a Version 5 Library

<table>
<thead>
<tr>
<th>Track</th>
<th>Member A</th>
<th>Member B</th>
<th>Member C</th>
<th>Member D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A A A A</td>
<td>B B B B</td>
<td>B C</td>
<td>D D D D</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>B B B B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>B B B B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>B B B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The DCB attributes of Version 5 SAS data libraries on IBM 3380s are DSORG=DA, RECFM=U, LRECL=32756, BLKSIZE=32760. The actual physical block size used when creating a data set is controlled by the BLKSIZE= option; the block size recorded in the DCB is set to the largest supported by the device type on which the data library resides. Since observations cannot span physical blocks, maximum observation length is controlled by block size. Therefore, in Version 5, the maximum observation length is 32k and that length is possible only on 3380 devices.

Version 6 Space Management Characteristics

The MVS SAS Version 6 data library implementation is much more device independent than that for Version 5. It is in effect a fixed block architecture (FBA) implementation on count-key-data (CKD) devices. A library is defined as a physical sequential (PS) data set with a fixed-standard (FS) record format and the block size and logical record length set equal to the physical block size to be used for the library. The physical block must be a multiple of 512 bytes. An example of a valid DCB specification is

DCB=(DSORG=PS,RECFM=FS,BLKSIZE=23040,LRECL=23040).

Observations are stored in pages which are an integral multiple of the physical block size. SAS data sets are stored in an integral number of pages. Therefore, the smallest unit of allocation for a Version 6 SAS data set is a page as opposed to a physical track for Version 5 data sets. Page size is not constrained by physical track size. Therefore observations may be considerably longer in Version 6 than in Version 5.
Page size may vary at the data set level within the library. Thus space utilization may be improved by using small page sizes (6k or less) for small data sets. (Clifford 1989) goes into considerable detail on page size considerations and the methods the SAS system uses to calculate the page size for a given data set when one is not specified.

The following example illustrates some of these points.

\[
\begin{align*}
\text{x} &= \text{pages in data set X (page spans 3 blocks)} \\
\text{y} &= \text{pages in data set Y (page span 2 blocks)} \\
\text{b} &= \text{physical blocks (7 per track)}
\end{align*}
\]

data sets X and Y each contain 4 pages

Track 1: x x x x x x x
   b b b b b b b
2 x x x x x y y
   b b b b b b b
3 y y y y y y y
   b b b b b b b

Backup, Restoration, and Migration Considerations

Historically, SAS data libraries have presented problems to generalized DASD management systems which provide data set archival, backup, restore, and migration functions. SAS Institute's official position is that SAS system utilities must be used for these purposes. In fact, we have worked with vendors of these products and strengthened our software to allow SAS data libraries to be managed by some of them.

One problem encountered was failure to honor track boundary constraints while restoring SAS data libraries. For example, a library might be restored as follows with blocks from multiple data sets residing on the same track.

\[
\begin{align*}
\text{Track}\quad &\quad \text{---------} \\
1\quad &\quad A A A A B B B \\
2\quad &\quad B B B B B B B \\
3\quad &\quad B B B B B B B \\
4\quad &\quad C C D D D D D
\end{align*}
\]

This problem was resolved by ensuring that SAS data libraries are marked with data set organization DA. As long as the DASD utility maintains track integrity for DA data sets, everything is fine.

A second problem was that it was not possible to use a SAS data library after restoration or migration to a different device type. A partial solution to this problem was provided in SAS Release 5.16. The original device type is stored in the library directory thereby allowing SAS software to recognize, for example, that a library originally created on an IBM 3330 had been restored to an IBM 3380. The restored library is subsequently managed in 3350 mode, that is, only the first 19k of each 3380 track is utilized. Of course PROC COPY can be used to create a copy of the library more suitable for 3380s. Even with this technique, it is still not possible to restore a library to a device with a smaller track size.

Neither of these problems exist when restoring or migrating SAS Version 6 data libraries. The only requirements made of DASD management software are 1) that it respect the RECFM=FS nature of the dataset, that is, do not attempt to block it and make it FBS, and 2) that it restore the library contiguously within the extents it allocates to it, placing the maximum number of physical blocks the device allows on each track. It is not necessary that the library be placed in a single extent. Even the standard IBM utility IEBGENER can be used to migrate a SAS Version 6 data library from one DASD type to another.

SAS DATA LIBRARY PERFORMANCE CHARACTERISTICS

Several aspects of SAS data library performance were studied. These included creating/reading data sets and directory management functions. For Version 6 data libraries, creating data sets in previously formatted library space and the effects of data compression were also examined. The data set creating/reading tests were performed on a data set containing 38,000 200-byte observations and occupying approximately 500 tracks on an IBM 3380. The code to create a data set is simply DATA statement with a DO loop; the code to read a data set is just DATA _NULL_; SET;

The metrics presented are elapsed time in seconds, I/O counts (EXCPs), and I/O connect time in seconds. I/O connect time is the time that the device is connected to the channel as measured by the MVS/XA channel subsystem. Minimizing I/O connect time is a very beneficial thing to do on mainframes because path contention is often a limiting performance factor on such systems.

Creating SAS Data Sets

Comparisons at optimal block sizes The first test compared the performance of the two library implementations using an optimal blocking factor for the 3380 device, two blocks per track. In the Version 6 case, page size is equal to block size.
The Version 8 result is better primarily because the Version 5 library, in keeping with the requirements for a true DA dataset, updates the track balance information in record 0 with each write. The difference in I/O counts is unexplained and will be investigated.

Comparisons at seven blocks per track. In the second test, the libraries were created with seven blocks (6k) per track. This block size is a common one in many installations because it fits nicely across a range of device types. Again, in the Version 6 case page size is equal to block size.

Effects of large page sizes. With the Version 5 data library, one block is the most that can be written in a single I/O operation. Since 23k is the largest practical block size for 3380 devices, this means that 23k is the largest amount of data that will be transferred in a single I/O. In Version 6, the unit of data transfer is a page. Therefore, much more data can be transferred in a single I/O. The next three tests explored creating data sets with different page sizes, each a multiple of an underlying 6k block size. These three tests are compared with an earlier one which used a half-track physical block size.

In this case, a large page size has a distinct performance benefit. Remember, though, that minimum space allocation for a data set is one page. (Bowman 1989) thoroughly covers page and block size selection considerations.

Reading SAS Data Sets

Comparisons at optimal block sizes and at 6k

Frankly, we do not know why the Version 5 results are better and plan to explore these cases further.

Effects of large page sizes. With the Version 5 data library, one block is the most that can be read in a single I/O operation. Since 23k is
the largest practical block size for 3380 devices, this means that 23k is the largest amount of data that will be read in a single I/O. In Version 6, the unit of data transfer is a page. Therefore, much more data can be read in a single I/O. The next three tests explored reading data sets with different page sizes, each a multiple of an underlying 6k block size.

### Effects of Varying Page Size ***

<table>
<thead>
<tr>
<th>Page Size</th>
<th>Elapsed Time</th>
<th>Read Time</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>24k</td>
<td>26s (1.7)</td>
<td>9.4s</td>
<td>979</td>
</tr>
<tr>
<td>42k</td>
<td>20s (1.3)</td>
<td>9.2s</td>
<td>580</td>
</tr>
<tr>
<td>84k</td>
<td>15s (1.0)</td>
<td>9.0s</td>
<td>309</td>
</tr>
</tbody>
</table>

Library Directory Management Performance

An area which has been weak with the Version 5 data library is the performance of data library directory management functions, especially when the directory is large. Two tests were run to compare several types of directory management activity on Version 5 and Version 6. The first test 1) creates 500 single observation data sets 2) runs a PROC CONTENTS on the library, and 3) uses PROC DATASETS to AGE the data sets. The second test uses PROC DATASETS to modify all the variable labels in an MXG® library PDB.

### Comparisons of V5 and V6 ***

<table>
<thead>
<tr>
<th>Directory Management Functions</th>
<th>Elapsed Time</th>
<th>CPU Time</th>
<th>I/O Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create data sets</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V5 6.2m (3.6) 42s (1.9) 28800 (2.8)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V6 2.9m (1.0) 22s (1.0) 10400 (1.0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>List contents</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V5 7.4m (14.8) 20s (2.5) 25300 (11.0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V6 0.5m (1.0) 8s (1.0) 2390 (1.0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Age data sets</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V5 5.6m (28.0) 13s (3.3) 19500 (24.4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V6 0.2m (1.0) 4s (1.0) 800 (1.0)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data Compression Tradeoffs

This final topic blends the two major themes of storage management characteristics and performance. Version 6 optionally performs data compression which may be requested globally or at the data set level. Intuitively, a data set containing many long character variables should be an excellent candidate for compression. As it turned out, one of our Data Center problem tracking data sets yielded excellent results. A more interesting situation is posed by data sets containing many numeric variables, such as MICS® or MXG performance data bases. Even these yielded surprisingly good results as can be seen in the following table. PROC COPY was used to copy an MXG daily PDB and an MICS detailed time span PDB to a Version 6 data library. The CPU time difference between PROC COPY with COMPRESS=YES and COMPRESS=NO and the size of each PDB were used to calculate a per megabyte data compression cost. The tests were run on an IBM 3084.

### Data Compression Tradeoffs ***

<table>
<thead>
<tr>
<th>Data Set</th>
<th>CPU Savings</th>
<th>Cost/MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>85%</td>
<td>0.61 cpu seconds</td>
</tr>
<tr>
<td>MICS</td>
<td>37%</td>
<td>0.78</td>
</tr>
<tr>
<td>MXG</td>
<td>28%</td>
<td>0.81</td>
</tr>
</tbody>
</table>

The overall CPU cost for data compression is likely to be minimal for many applications. An estimate based on the above results indicates, for example, that use of data compression would add less than three percent additional CPU time to an MXG daily BUILDPDB job.

Whether or not data compression is worthwhile to you depends partially on the cost allocation policy your data center uses in calculating charges. However other factors can also be important. If data compression is very effective, as in the case of the internal application noted above, decompression costs are considerably less than compression costs. Application elapsed times can also be significantly reduced because fewer I/Os are needed to process a compressed data set.

**SUMMARY**

We have compared some storage management and performance characteristics of Version 5 and Version 6 SAS data libraries under MVS and seen that the Version 6 implementation offers significant advantages which include:

- The highest degree of device independence possible with CKD devices.
- Potentially significant space reductions, especially in libraries containing many small members.
- Easier data library backup, restoration, and migration.
- Significant performance improvements in data set creation, reading, and directory management in many situations.

Please note that this paper covers only very specific aspects of SAS data library performance and that many other factors contribute to the total performance of SAS software in your applications. Therefore it is not valid to infer solely from the results presented here that your applications will perform markedly better under SAS Version 6 than under SAS Version 5.

References:


SAS is a registered trademark of SAS Institute Inc. Cary, NC. USA

MICS is a trademark of Morino Associates Inc., Vienna, VA. USA

MXG is a registered trademark of Merrill Consultants, Dallas, TX. USA