Performance Issues to Consider When Migrating Production Jobs from Version 5 to Version 6 of the SAS® System under MVS

Amitava Ghosh and Leigh A. Ihnen, SAS Institute Inc.
Cary, NC

ABSTRACT

This paper deals with performance issues when migrating production jobs from Version 5 to Version 6 of the SAS® System under MVS. Due to the significant differences between the architectures of Version 6 and Version 5 of the SAS System, the user might experience performance degradation/improvement when migrating some jobs to Version 6 of the SAS System. Specific areas to consider are formats and informats, macro processing, SAS DATA step, and so on. This paper will try to highlight some of the known performance problem and improvement areas. It also suggests possible workarounds or alternate methods of coding the SAS program to overcome some of the performance degradation.

INTRODUCTION

Version 6 of the SAS System is built on the concept of MVA ™ (MultiVendor Architecture). MVA allowed the SAS System to be designed in a portable manner, which meant most of the executable code is hardware and software platform independent; while only a small portion of the total executable code is hardware and operating system dependent. Because most of the code is portable, the look and feel of the SAS System is essentially the same in all the different hardware platforms on which the SAS System is supported.

On the other hand, Version 5 of the SAS System evolved through the years primarily on the IBM® mainframes. It was designed in most part specifically to take full advantage of the IBM 370 architecture and hence it performed remarkably well in terms of its execution speed. It was mostly written in IBM assembler or PL/I languages. The downside of doing this was that any major changes to the underlying hardware or operating system required considerable changes to the internal SAS executable code.

Because Version 6 of the SAS System was designed with very little dependence on the underlying hardware and software, it is possible to essentially port the SAS System from one vendor hardware platform to another with very few changes. Because most of the executable code have been written in ANSI compliant C, there is very little problem in porting internal C code from one operating system to another. This is good for the SAS user; the SAS program he or she writes will run on all the platforms on which the SAS system is supported. The only negative element of MVA is that in some areas, significant performance degradation occured when compared to Version 5 of the SAS System. This was primarily due to the fact that Version 6 introduced a lot more function layers, did not take advantage of some of the capabilities available on MVS, and was not written in assembler.

Since the introduction of Release 6.06 of the SAS System, many changes have been introduced to the architecture to make the SAS System perform just as fast as Version 5 and in many cases significantly reduced execution and elasped time. However, there are still some areas of the SAS System that are still seeing some performance degradation relative to Version 5 of the SAS System.

MIGRATION ISSUES

There are some issues that a user must consider when migrating production SAS jobs from

Version 5 to Version 6 of the SAS System. Whenever the underlying architecture of the software is changed, the performance characteristics of the production jobs will also change. Is it possible to predict what types of jobs will perform better in Version 6 relative to Version 5? Can significant performance gain be obtained by recoding critical sections of the SAS job? What implicit assumptions about software behaviour have been used in defining existing production software? This paper presents several rules of thumb that can be used to estimate the effects of migrating from Version 5 to Version 6 of the SAS System.

Which is slower?

- macro variable resolution.
- Parsing of SAS code
- Proc FORMAT style formats
- some Formats and Informats
- Resolution of SAS variables against data sets
- Printing

Which is faster?

- SAS I/O
- Extemal file I/O
- DATA step logic constructs
- Sorting
- Procedures

Which is new?

- SORTEDBY data set option
- DATA step Views
- Saved DATA step programs
- SAS Institute supported versions of contributed procedures

Which is missing or obsolete?

- User exits
- SUGI supplemental library
- Version 5 user-written procedures

Macro variable resolution

Due to architecture changes that were made in Version 6 of the SAS System, jobs that use SAS Macro Language might see an increase in execution time. To help in reducing execution time of jobs, several new system options have been added. MSYMTABMAX is an option that controls the size of the incore macro symbol table. When the incore symbol table is filled, the SAS System will start writing the macro symbols out to disk, which should increase execution time. MVARSIZE is an option that controls the size of a macro variable which will be kept in the incore symbol table. Any macro variable whose size is larger than MVARSIZE will be written to disk. To help in increasing performance of macro intensive jobs, a new facility called compiled macro stored facility has been added to the SAS Macro Language. This facility enables a user to compile and store a macro in disk and retrieve it any time the macro is needed. The advantage of stored compiled macro is that the macro in question has to be compiled only once and not every time it is first referenced in a SAS job. Obviously this is only an issue if the compile time for the macro is significantly long.

Parsing of SAS code

In Version 6, multiple source code streams can be generated. SCL code can push source statements, statements can be included using the %INCLUDE statement from the Program Editor and the command line, and the Program Editor can submit statements. The changes in input stream handling require more logic than Version 5. Jobs that consist mainly of many lines of SAS code with very little data and processing will require more CPU time than Version 5. A well commented program can be sped up by removing the comments.

PROC FORMAT style formats

In Version 5, formats generated by PROC FORMAT are load modules. This allowed users to build format libraries as PDSs and concatenate PDSs in JCL. In Version 6, PROC FORMAT style formats are SAS data objects stored in SAS catalogs. In Version 6 it is not possible to concatenate SAS catalogs like MVS data sets in JCL. It is possible to specify a format search list in the CONFIG file. Because Version 5 style formats support shorter labels than Version 6 formats and are load modules, they generally have faster performance than Version 6 formats. If performance or concatenation of format libraries are issues, then existing Version 5 formats can be used with Version 6.

SAS formats and informats

With Version 6, a portable code generator has been implemented to support the SAS DATA step. Also many SAS formats and informats have been converted from assembler to portable C code. As a result, most formats and informats are slightly slower than Version 5. Papers presented at previous SUGI conferences have presented tables of relative performance.

Resolution of SAS variables against SAS data sets

In Version 5, a binary search was used to lookup variables on a SAS data set. Version 6 uses a linear search to resolve variable names. For procedures like PROC MEANS and PROC SUMMARY that can produce a large number of output variables, performance can be severely degraded.

Printing

In Version 5, formatting of numbers zeroed some of the low order bits. Version 6 attempts to produce the most accurate representation without suffering severe performance penalties. Version 6 also supports national language output through message files. These two changes cause procedures like PROC FREQ to be slower than Version 5 when large amounts of output are produced.

SAS I/O

With Version 6, the SAS System has incorporated a new format for SAS DATA Libraries. In Version 5, SAS data libraries were Direct Access (DA) data sets. This meant that these data sets could not be easily moved from one type of direct access device to another that had a different track capacity, because DA data sets are track dependent. Also, there was a 4 byte overhead for each observation in a SAS data library. Version 6 changed the format from DA to Fixed Standard(FS). Because a FS data set is just a sequential file, it is not track dependent and you can use any MVS copy utility like IEBGENER to copy SAS data libraries from one data volume to another regardless of its track capacity. Secondly, Version 6 reduced the overhead from 4 bytes per observation to 1 bit per observation. Because Version 6 has significantly less overhead per observation, users will see a major reduction in execution time for jobs that are SAS I/O intensive.

Along with this new format for SAS DATA Library, Version 6 introduced the concept of locking either at SAS member level or at record level. Member level implies that only one user can open a member for update at any particular time. Record level locking implies that only the current record being updated is locked. It must be pointed out that record level locking is relatively quite expensive in terms of CPU usage and hence must be carefully used. Also , by default, any files opened for update have record level locking. PROC APPEND is one which always opens files for update.

External file I/O

External file I/O in Version 6 is considerably faster than Version 5. One of the main reasons for this is that Version 6 uses BSAM access method rather than QSAM as its was in Version 5. Because BSAM does block I/O, significant performance improvement can be obtained by choosing an optimum blocksize for that device.

DATA step logic constructs

With Release 6.07 the DATA step applies boolean short circuiting to the evaluation of logical expressions. The major implication is the point at which use of PROC FORMAT style format in a subsetting IF statement using the PUT function is faster than an IF statement with

boolean expression containing all selection criteria. In Version 5 the switch over point is about 20 subclauses. Version 6 requires about 60 subclauses.

Procedures

Many of the numerically intensive procedures are faster in Version 6. PROC GLM, FACTOR and many other linear algebra based procedures also benefit from the availability of a vector facility on IBM hardware. In small sample cases Version 6 might be slower than Version 5. Procedures such as PROC SUMMARY also benefit from a larger address space. If all the data can be processed in memory PROC SUMMARY can be significantly faster in Version 6. For very large cross classifications on MVS, two undocumented options that can improve performance are VMNSISA and VMNSOSA. Suggested values are VMNSISA=32K and VMNSOSA=32K.

Sorting

In Version 6, the SAS sort has been greatly improved. For small data sets, SAS sort is competitive or faster than the HOST sort. To improve SORT performance, the UBUFSIZE option can be increased to a multiple of 6K that is between 32K and 64K.

Filename statement

Version 6 introduced a new SAS statement called FILENAME. This performs most of functions that are available via the TSO ALLOCATE command and JCL DD statement. In other words, one might use the FILENAME statement to dynamically allocate and create MVS data sets. Because FILENAME uses SVC 99 services provided by MVS, this facility works in both batch and TSO environment.

SORTEDBY data set option

A new SAS data set option SORTEDBY can be used to assert how the SAS data set is sorted and hence be used to prevent redundant sorts. This is especially useful when the input data used to create the SAS data set are already sorted. This sortedby value is stored as part of the data set.

DATA step Views

One of the major enhancements of Version 6 over Version 5 is the DATA step View. By creating a DATA step View, you are essentially specifying a DATA step to be a SAS I/O engine and that can be used directly by a SAS procedure which requires an input SAS data set. Essentially, each iteration of a DATA step will yield a SAS observation that will be inputted directly into the SAS procedures. To do the same in Version 5 required one to execute a DATA step to create an intermediate SAS data set. By using DATA step Views, one does not have to materialize a copy of the input data into a SAS data set, hence one can save disk space. Obviously , this is an issue only if the input data set is very large and the available disk space is limited. It must be pointed out that in Release 6.08, the DATA step View is considerably faster than its previous version in Release 6.07 of the SAS System.

Institute supported SUGI Supplemental procedures

- COXREGR: The functionality of this procedure is included in the PHREG procedure in Release 6.07 of the SAS/STAT® software.
- ALSCAL: The MDS procedure in Release 6.07 includes much of the functionality of this procedure.
- FMTLIB: This is now part of the FORMAT procedure.
- LOGIST: The functionality of this procedure is included in the LOGISTIC procedure distributed in Release 6.04 and Release 6.06 of SAS/STAT software. Much of the LOGISTIC procedure syntax is similiar to that of the LOGIST procedure.

User Exits

In Version 6, many of the SASUSER exits available in Version 5 are currently not yet implemented. Also, the ones that do exist are different in their implementation and hence will require the user to rewrite them. Version 5 style infile-file exits are available in Version 6;, however, the interface API is different, hence user infile-file

exits must be rewritten. Support for all of the user exits in Version 5 should be available in a future SAS release.

Version 5 user-written procedures

Version 5 style user-written procedures are not supported in Version 6. However, users may write Version 6 style procedures by using SAS/TOOLKIT® software.

CONCLUSION

In conclusion, there are a few issues a user must address to make sure that Version 5 jobs meet performance expectations in Version 6. A few things are basically slower in Version 6 and will require users to recode their SAS jobs in a manner that utilizes new capabilities of Version 6 to increase performance. However, most features in Version 6 perform just as fast as they did in Version 5 and users should take full advantage of them wherever possible. It must be pointed out that Version 6 of the SAS System under MVS exploits MVS/XA® and MVS/ESA® features provided by MVS. SAS jobs in Version 6 running under MVS/XA or MVS/ESA environment, have access to much larger memory regions; hence significant performance gains can be obtained by increasing the size of the SAS job region.

TRADEMARKS