

DATA Step Performance

Pat M. Herbert, SAS Institute Inc., Cary, NC
Jeffrey A. Polzin, SAS Institute Inc., Cary, NC

ABSTRACT

For production sites, several changes in Release 6.07 will help reduce DATA step CPU usage. One of the first concerns in producing Release 6.07 was SAS data set I/O because the vast majority of DATA steps produce SAS® data sets and a large number of them also read SAS data sets. Another major concern was INFILE performance. From tests on Release 6.06, it was clear that Release 6.06 unformatted external file input was faster than Release 5.18 external input. This led to benchmarking of all informats, which were one source of poor performance. Another source of poor INFILE performance was column positioning. In Release 6.07, DATA step column positioning has been improved, especially positioning expressions such as `offset +5`, `+5`, and `@5`. Similarly, informats have been improved by up to 700%. Special attention has also been paid to DATA step functions. Performance has also improved when output variables are being created and when the number of variables defined in the DATA step is large. In general, data set creation is substantially improved over Release 6.06.

INTRODUCTION

One of the primary goals for Release 6.07 of the SAS® System was to improve overall performance. While working toward this goal, a major emphasis was placed on DATA step performance characteristics, with traditional uses getting the most attention. This has resulted in numerous improvements in CPU utilization, memory usage, and I/O throughput, many of which helped not only the DATA step but also procedures and the SAS System as a whole. These improvements will be especially welcomed by the mainframe user community, which was impacted the most by Version 6. In order to achieve the necessary improvements, a very wide range of changes had to be made. All of the changes combined should provide substantial performance gains for almost any application. Some of the enhancements that provide improvements for the DATA step include SAS data set I/O, flat file I/O, character-to-numeric and numeric-to-character conversions, numerous formats and informats, DATA step functions, and many others.

Many of the changes discussed also impact other areas of the SAS System. For instance, improvements in SAS data set I/O will also offer gains to procedures that process substantial amounts of data. Additional areas concerning performance of the SAS System to be discussed at SUGI 17 include a paper on procedure performance gains given by Leigh Ihnen and a paper on I/O performance presented by Steve Beatrous.

The CPU time statistics provided here were obtained on an MVS operating system running on an IBM® 3090-600S. Similar results should be realized on other systems. It should also be noted that performance measurement of a product as large as the SAS System or even just of the DATA step is not a trivial exercise. There are many features and functions that affect how the system as a whole performs. The use of (or removal of) any particular option or feature from a DATA step can change its performance characteristics. An attempt was made to use "typical" constructs in the test programs, but you should be aware that even the values within the data can impact the operation of a DATA step. With that said, let's look at some specific improvements that can be expected with Release 6.07.

SAS I/O

SAS data set I/O has been improved dramatically. Efforts were concentrated in this area because typical SAS applications tend to be I/O intensive. Improvements in SAS data set I/O account for major reductions in CPU usage as well as reducing disk space requirements and I/O counts. These gains are the result of use of the new lean file format as well as path length reductions provided by fast pathing of I/O function calls. There was also an effort to tune the system appropriately for each host machine and then to use these results to set the defaults for the shipped systems. All of this combined results in SAS I/O now doing fewer real I/Os and consuming less CPU time than Release 6.06. In fact, it outperforms Release 5.18 as well.

The new lean file format, which is the default for Release 6.07, provides better performance because it stores observations in a SAS data set with less overhead than previous releases. This file format has a 1 bit per observation overhead, while Release 6.06 had 12 bytes of information saved with each observation and Release 5.18 used 4 bytes. For SAS data sets with large numbers of observations and few/small variables, the lean file format can reduce the disk space used by more than 50%. This change also accounts for some of the CPU savings because less data have to be moved when an observation is read or written.

Fast pathing of many of the I/O functions reduced the path length necessary to accomplish their work and this in turn resulted in better CPU utilization. This technique is especially useful for sequential processing, which is how many DATA steps access the SAS I/O system. Basically, when an observation is returned to a caller, if the next observation is currently available in an I/O buffer, the function address that is used by the caller is modified to point to an optimized routine that simply returns the in-memory data as needed. When the next observation is not available in memory, the function address is changed to point to an I/O routine that will read the data page into memory that contains the next set of observations.

SAS data set I/O was extensively tuned on each host in order to determine the optimum settings for options. The results of these tuning efforts were then applied to the defaults that are shipped with the system so that a newly installed system will exhibit favorable performance characteristics. These options take into account the need to minimize memory consumption, while at the same time trying to optimize SAS data set I/O throughput and CPU usage.

Examples of performance gains with respect to sequential SAS input can be demonstrated with the SET, MERGE, and UPDATE statements within a DATA step. Similarly, the OUTPUT statement can be used to show improvements for SAS output. The POINT= option used with the SET statement has also been enhanced in Release 6.07. (Even with this improvement, however, it is still somewhat less efficient than it was in Release 5.18 and still has some room for improvement.) The following table was produced by collecting timing information taken by processing 100,000 observations with 20 numeric variables on an MVS operating system.

CPU time in seconds processing 100,000
observations with 20 numeric variables

Type of SAS I/O	5.18 CPU time	6.06 CPU time	6.07 CPU time	Inc/(Dec) vs. 5.18	Inc/(Dec) vs. 6.06
SET statement	1.62	1.74	0.90	(80%)	(93%)
MERGE statement	2.79	2.63	1.81	(54%)	(45%)
UPDATE statement	4.31	5.11	3.51	(22%)	(45%)
OUTPUT statement	1.29	1.41	0.83	(55%)	(69%)
SET with POINT=	1.43	2.01	1.82	27%	(10%)

Another way to enhance performance with respect to SAS data set I/O is to use the MODIFY statement which is new with Release 6.07. This statement can be used in many cases rather than using the UPDATE statement and allows users to alter SAS data sets in place, whereas the UPDATE statement produces an entirely new output data set. This can result in huge reductions of real I/Os and less CPU usage as well.

INFILE AND FILE I/O

Because most SAS data sets are produced by reading external files, this part of the DATA step received a great deal of attention, too. The primary way that the DATA step accesses external files is through the FILE/PUT and INFILE/INPUT statement groups. These were the next areas to get significant attention. The same technique used for SAS data set I/O to reduce execution path length had been used successfully in Release 6.06 for external file input. In Release 6.07, this technique was applied to file output and performance characteristics there approached those of Release 5.18.

This enhancement is made evident in an example where 100,000 records of 80 bytes in length were processed. The CPU time used to read these records with the INFILE and INPUT statements is significantly less in Release 6.06 than it was in Release 5.18. Likewise, the FILE and PUT statements used for external file output were improved in Release 6.07. Another factor that alters the results for external output is the use of the PRINT option in the FILE statement. The results using this variation have also been provided as they show an area that can be improved upon in a future release.

CPU time in seconds processing 100,000
records of 80 bytes in length

Type of external I/O	5.18 CPU time	6.06 CPU time	6.07 CPU time	Inc/(Dec) vs. 5.18	Inc/(Dec) vs. 6.06
INFILE/INPUT	1.54	1.21	1.18	(30%)	(2%)
FILE NOPRINT/PUT	1.72	2.16	1.74	1%	(24%)
FILE PRINT/PUT	1.87	2.70	2.25	20%	(20%)

In Release 6.07, DATA step column positioning has also been improved. INPUT statements using constructs like *offset*+5, +5 and @5 have been improved by approximately 6% and are now roughly equivalent to Release 5.18 with respect to CPU utilization.

FORMATS AND INFORMATS

Because formats and informats are typically used in conjunction with FILE and INFILE processing, this was the next area to be attacked. Particular attention was given to character-to-numeric informats and numeric-to-character formats. These include informats like 8.2, E8, and COMMA10.2, and formats like BEST12, as well as the w.d format. Along with these, more than 100 formats and informats were studied and modified, some with extremely positive results.

In the area of numeric-to-character and character-to-numeric conversion, the class of value being converted was a controlling factor. The most significant improvements were made for values without fractional components. These integer values are now processed in an optimized way which reduces the CPU usage for these values. Conversion of values with fractional components was also enhanced but to a slightly lesser degree. Because a great many internal routines access this set of formats and informats, the improvements made here indirectly impact many other areas for a net positive effect on system performance. The following table illustrates the gains made in numeric-to-character and character-to-numeric formats and informats.

CPU time in seconds for various char-to-numeric/numeric-to-char
informats/formats using integer and floating point data

Type of conversion	5.18 CPU time	6.06 CPU time	6.07 CPU time	Inc/(Dec) vs. 5.18	Inc/(Dec) vs. 6.06
informat (int)	1.12	1.35	0.98	(14%)	(37%)
informat (float)	1.64	1.72	1.35	(21%)	(27%)
format (int)	0.89	2.24	1.35	51%	(66%)
format (float)	1.57	2.92	2.73	73%	(1%)

There are numerous other formats and informats that were also improved with respect to CPU usage. In the informat category, DATETIME uses four times less CPU resources than it did in Release 6.06. Similarly, the DATE, TIME, \$HEX, HEX, E, PD, MONYY, NENG0, YYMMDD, and SMFSTAMP informats improved dramatically. With respect to formats, BEST, YYMMDD, YEAR, MONTH, YYQ, IB, PIB, PD, WEEKDAY, DAY, and many others also exhibit significant improvements. The following table illustrates the amount of change made to many of these and other formats and informats.

CPU time in seconds for various informats

Type of conversion	5.18 CPU time	6.06 CPU time	6.07 CPU time	Inc/(Dec) vs. 5.18	Inc/(Dec) vs. 6.06
NENG032.	.	1.77	1.27	.	(39%)
E8.	1.24	1.30	1.08	(15%)	(20%)
8.3	1.24	1.33	1.09	(14%)	(22%)
PD4.	0.91	1.29	1.09	20%	(18%)
B88.	0.91	1.29	1.10	21%	(17%)
SMFSTAMP8.	0.82	1.37	1.07	30%	(28%)
RMFSTAMP8.	0.76	1.41	1.12	47%	(26%)
DATETIME32.	0.63	4.94	1.23	95%	(302%)
DATE10.	0.78	2.75	1.24	59%	(122%)
TIME8.	0.62	2.58	1.20	94%	(115%)
\$HEX16.	0.45	1.14	0.83	84%	(37%)
YYMMDD8.	0.70	1.73	1.30	86%	(33%)
MMDDYY8.	0.69	1.66	1.25	81%	(33%)
DDMMYY8.	0.63	1.53	1.16	84%	(32%)

CPU time in seconds for various formats

Type of conversion	5.18 CPU time	6.06 CPU time	6.07 CPU time	Inc/(Dec) vs. 5.18	Inc/(Dec) vs. 6.06
84.	2.26	1.26	1.20	(88%)	(5%)
32.15	1.62	1.23	1.12	(45%)	(10%)
IB4.	1.02	1.49	1.13	11%	(32%)
PIB8.	1.02	1.49	1.13	11%	(32%)
RB8.	0.91	1.50	1.12	23%	(34%)
MONYY7.	0.96	1.44	1.19	24%	(21%)
FRACT32.	0.93	1.51	1.20	29%	(26%)
ED4.	0.87	1.36	1.13	30%	(20%)
\$OCTAL15.	.	5.35	1.11	.	(382%)
BEST12.	0.68	1.94	1.13	66%	(72%)
YYMMDD8.	.	1.87	1.37	.	(36%)
YEAR4.	.	1.52	1.12	.	(36%)
YYMMDD8.	.	1.58	1.17	.	(35%)
MONTH3.	.	1.52	1.13	.	(35%)
YYQRM10.	.	1.60	1.19	.	(34%)
YYQ8.	.	1.57	1.17	.	(34%)

Although most of the informats and formats are much better than in Release 6.06, there are still some areas needing further improvement.

DATA STEP FUNCTIONS

Another area where performance improvements can be demonstrated is with DATA step functions. There are over 95 functions that now use significantly less CPU time than they did in Release 6.06 and more than 50 functions are better than their Release 5.18 counterparts. Again, emphasis was placed on the more traditional functions like SUBSTR, UPCASE, LEFT, VERIFY, SUM, MIN, MAX, FLOOR, and CEIL. Many of these functions are also used internally, which means these gains show up in places other than just the DATA step. These reductions came in the form of optimized call sequences for whole groups of functions, particularly those with a single argument, as well as algorithmic enhancements to the individual functions. Another technique used to improve functions was used on the LAG function - this function is now part of the machine code that is generated by the DATA step when it is compiled. A selected list of improved functions, along with information about the amount of improvement, is shown in the following table:

CPU time in seconds for various DATA step functions					
Function	5.18 CPU time	6.06 CPU time	6.07 CPU time	Inc/(Dec) vs. 5.18	Inc/(Dec) vs. 6.06
UPCASE	16.41	1.21	1.05	(183%)	(15%)
DATETIME	6.33	12.03	1.04	(509%)	(123%)
DATE	6.26	6.09	1.07	(485%)	(469%)
TODAY	5.95	5.82	1.03	(478%)	(465%)
TIME	3.92	4.39	1.03	(281%)	(326%)
LAG	1.87	2.98	1.03	(82%)	(189%)
SUM	1.70	1.88	1.07	(59%)	(76%)
LEFT	1.55	2.97	1.06	(46%)	(180%)
MIN	0.93	1.09	0.72	(29%)	(51%)
PIFSTATE	1.34	1.21	1.09	(23%)	(11%)
MAX	0.91	1.08	0.75	(21%)	(44%)
VERIFY	0.94	7.53	1.06	13%	(610%)
JULDATE	0.92	6.17	1.10	20%	(461%)
TRANSLATE	1.28	5.49	1.11	(15%)	(395%)
INDEXC	0.97	3.31	1.06	9%	(212%)
FLOOR	1.21	2.07	1.05	(15%)	(97%)
CEIL	1.20	2.06	1.06	(13%)	(94%)

It should be noted that a flaw in the implementation of the Release 5.18 UPCASE function resulted in the use of an inordinate amount of CPU time.

MISSING VALUE DETECTION

Another area that was improved was where warning messages are issued when a function returns a missing value (usually as a result of an operation on a missing value). This is particularly important when the warning situation occurs many times. In this case, the DATA step keeps a count of the number of occurrences and the line and column position where the warning was issued. In an example where this type of warning occurred 200,000 times, the program used 1.95 CPU seconds in Release 6.06 and only 1.30 CPU seconds in Release 6.07 for a 50% reduction in resources used. The technique used to improve this area was the way that the DATA step generated code and the runtime functions used to support missing value detection. In Release 6.06, the technique used was to perform a binary search over a table containing statement locators and their corresponding memory addresses in the generated code. The binary search was noticed to be very expensive with regard to machine resources. In Release 6.07, the methodology was modified based upon the implementation used in Release 5.18. The modified implementation generates code, at appropriate points in expression generation, that keeps track of the expression missing

value detection counters, so that as the code is being executed a global pointer to the current counter is always available for use by the runtime support routines. A further refinement was introduced, using the DATA step option ERRORBLOCK, which keeps track of the expression missing value detection counter for each expression compiled in a DATA step program and not all of the subexpressions in the expression.

ARRAYS

Explicitly subscripted arrays within the DATA step have also been optimized. This optimization is especially apparent when the subscript is a constant value. In these cases, the array references are pre-evaluated when the DATA step is compiled, yielding CPU use that is roughly one-third what was used in Release 5.18. Also, new with Version 6 of the DATA step is support for multidimensional explicitly subscripted arrays. Use of implicitly subscripted arrays is not recommended as their performance characteristics were not improved and they tend to use much more CPU resources than explicitly subscripted arrays.

DO LOOPS

Even simple DO loops have been optimized. All three forms of loops (iterative, WHILE, UNTIL) exhibit improvements in CPU utilization. Although it would require an enormous number of loop iterations to achieve a noticeable change in an application, improvements to this area exemplifies the effort given to decreasing the CPU requirements within the DATA step.

CPU time in seconds for various DATA step loop constructs doing 3,000,000 iterations					
Type of Loop	5.18 CPU time	6.06 CPU time	6.07 CPU time	Inc/(Dec) vs. 5.18	Inc/(Dec) vs. 6.06
Iterative	1.99	1.50	1.05	(89%)	(42%)
DO-WHILE	3.80	1.87	1.32	(187%)	(41%)
DO-UNTIL	4.25	1.78	1.23	(245%)	(44%)

BOOLEAN EXPRESSION HANDLING

The DATA step has also seen significant improvements in the area of Boolean expression handling. This change will impact applications where multiple expressions are bound together in a statement (for example, an IF statement) with OR or AND operands used to connect the expressions. An example of such a construct is demonstrated in the following statement:

```
IF VAR1 OR VAR2 OR VAR3 OR VAR4 THEN PUT 'One of these is true';
```

The idea behind such short circuiting is that, in a statement like the one above, the first true value results in the entire statement being true and thus additional tests are not necessary. The following table demonstrates the large CPU use reductions:

CPU time in seconds for random true/false determination with Boolean short circuiting					
Type of Expression	5.18 CPU time	6.06 CPU time	6.07 CPU time	Inc/(Dec) vs. 5.18	Inc/(Dec) vs. 6.06
5 OR expressions	6.17	8.05	1.13	(446%)	(612%)
5 AND expressions	5.34	7.91	0.82	(551%)	(864%)

BIT STRING TESTING

In many cases, bit string testing is now generated by the DATA step compiler rather than using a function call to do the test. This change applies to single character tests as well as numeric value tests. Unfortunately, use of bit string testing on character strings with a length greater than one still exhibits a degradation.

CPU time in seconds for
800,000 bit string operations

Data type	5.18 CPU time	6.06 CPU time	6.07 CPU time	Inc/(Dec) vs. 5.18	Inc/(Dec) vs. 6.06
char1	1.22	2.19	0.87	(40%)	(151%)
char2	1.73	16.52	16.31	842%	(1%)
char4	1.74	11.85	11.65	569%	(1%)
int1	1.23	2.73	1.63	32%	(67%)
int4	1.23	2.73	1.65	34%	(65%)

GENERAL CODING CONSTRUCTS

In the mainframe environment, a technique that was used to enhance the performance of the DATA step was use of routines that do not require dynamic save areas (DSAs). These routines are more efficient because they do not require save areas to be acquired when they start or to be freed when they terminate. In order to make some routines have the quality of not requiring a dynamic save area, they often had to be modified such that called routines were relocated into higher level routines and, in some cases, use of automatic storage had to be reduced. Source code modifications were also made in some cases to bring small functions in line rather than calling them at all. Also, in the mainframe environment, many of the specialized functions were optimized. These improvements came in the form of reduced path length for the most common uses, better use of the available instruction set, and removal of duplicate validation checks.

CODE GENERATION

There have been additional optimizations to the DATA step compiler where the actual machine instructions are generated. These changes provide for smaller generated machine code programs or compiled programs and less CPU overhead.

NEW FEATURES

Additionally, there are many new features that can also enhance DATA step performance. One new feature is the *stored program facility* which allows DATA steps to be compiled into an intermediate form and saved for later execution. This could be especially beneficial where large DATA step programs are used because the compile phase can be bypassed. Similarly, DATA step views can be pre-compiled and they also have the additional benefit of not requiring that the data to be processed by a subsequent procedure be stored in an intermediate SAS data set if it is not necessary.

OPTIONS

There are also numerous tuning options (both SAS System options and DATA step options) that can be used to adjust the performance characteristics of the DATA step. Many of the DATA step options should only be used with production applications that have been thoroughly tested. A brief list of some of these options follows:

SYSTEM OPTIONS

- BUFNO= Number of buffers to use for each SAS data set

- BUFOBS= Number of observations per buffer
- BUFSIZE= Size of internal I/O buffer
- COMPRESS= Compress observations in created data sets?
- ENGINE= Default access method for SAS libraries
- MEMSIZE= Set the total amount of dynamic memory available
- NOSTIMER Print performance statistics after each step?

DATA STEP OPTIONS

- ERRORBLOCK Group error and warning messages to entire expressions
- NOMISS Skip checks for missing values
- NOSTMTID Do not maintain statement number information for use with error messages
- NOSUBCHK Skip checks for array subscripts

DATA step options can be specified by following the DATA statement with a / (forward slash) followed by the option names, as follows:

```
DATA TESTOPT/NOSTMTID NOSUBCHK;
```

CONCLUSION

Performance of the SAS System as a whole was an important issue in the development process for Release 6.07. Because the DATA step is traditionally the single most used part of the SAS System, attention was focused on it. Many specific areas were studied and then modified or completely rewritten with this in mind. All of these changes, as well as others, have resulted in overall reductions in CPU utilization in the DATA step. Also, many of the changes intended to improve the DATA step have produced gains for procedures and other parts of the SAS System. The net result should be that users of Release 6.07 should realize savings in almost all production SAS programs over Release 6.06 and, in some cases, even over Release 5.18.

A final note: SAS Institute is dedicated to producing quality and efficient products. If you are aware of specific problems with respect to performance, please do not hesitate to bring this to our attention so that the appropriate action can be taken for future product releases.

ACKNOWLEDGMENTS

The information contained here is the result of countless hours by many individuals spent researching the areas that most needed improvement and then making the necessary changes to allow the DATA step and the rest of the SAS System to perform adequately. Among those who have contributed to this massive effort are Robert Cross, Leigh Ihnen, Amitava Ghosh, Oliver Bradley, Jack Wallace, Steve Beatrous, and others too numerous to name but no less important to the work that has been done. It should be noted that this was a companywide effort and should be recognized as such. We should also recognize the important role that the users have played in locating and pointing out many of the areas that needed attention.

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. * indicates USA registration.

IBM is a registered trademark of International Business Machines Corporation.

Other brand and product names are registered trademarks or trademarks of their respective companies.