Paper 9-25

Version 8 Base SAS® Performance: How Does It Stack-Up?

Robert Ray, SAS Institute Inc, Cary, NC

ABSTRACT

This paper presents the results of a study conducted at SAS Institute Inc to compare the time-domain performance of several key parts of the Base SAS® system at Version 8 and Version 6. The performance of the DATA Step, the SORT procedure, the SUMMARY procedure and the SQL procedure are compared on four platforms: OS/390, Windows NT, Solaris and HP-UX. Although the focus of this study is non-printing applications, some OSD related tests are presented for the DATA Step and the TABULATE procedure. Special attention is given to changes in PROC SORT and PROC SUMMARY that improve their performance over Version 6.

INTRODUCTION

The purpose of this paper is to present an overview of the results of a study to evaluate the time-domain performance characteristics of several key elements of the Version 8 Base SAS® System as compared to Version 6. This study looks at the DATA Step, and the SORT, SQL and SUMMARY procedures on four different platforms: OS/390 (MVS), Windows NT (WIN), Solaris (SLX) and HP-UX (H8X). Although it is impossible to precisely predict performance that our users will experience, the tests presented should give a general indication of the performance characteristics of Version 8 relative to Version 6. The tests conducted focus is on I/O and computationally intensive tasks with little or no printing since this type of work is likely to dominant computing resources. A few non-printing tests were included to indicate the general overhead of the new ODS printing subsystem.

TESTING STRATEGY

The developers currently responsible for each tested product designed the tests for their respective products. In most cases, individual tests were designed to evaluate specific aspects of performance. For example PROC SORT was tested against identical data sets with various initial orderings. PROC SQL tested joines of the same tables using different join techniques. While PROC SUMMARY included tests to isolate classification overhead. Where applicable, a given test was performed over a broad range of data set sizes to evaluate scalability factors.

To gather timing data from the tests, four host machines were each loaded with all versions of the SAS System to be tested. CPU or elapse times were extracted directly from the SAS logs. All test hosts except MVS were used exclusively for performance testing. Tests were run repeatedly to ensure repeatable results. CPU times were recorded for all hosts except Windows NT where elapse times were used since Version 6 for that host does not provide CPU times in the SAS Log.

RESULTS PRESENTATION

Because we are only interested in *relative performance*, all timing results have been reduced to unit-less ratios. All plotted points have be scaled either by the time of the equivalent Version 6 test point or by the maximum time value for all points of a given test. This prevents any possibility of comparison of test across hosts since that would be highly hardware dependent. We in no way intend these results to be used to judge the relative performance of the SAS System between the tested hosts.

To present summary data, *box-and-whiskers* plots generated by the SHEWHART procedure have been used. Definition of the plot elements can be found in figure 1 below. Note that an open box symbol at either end of the *whiskers* (vertical line) indicates one or more clipped values.

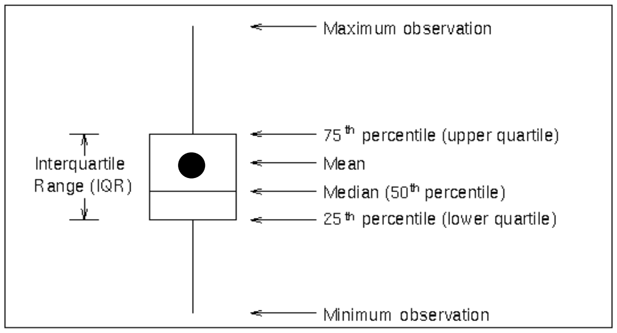
Maximum observation
75th percentile (upper quartile)
Mean
Median (50th percentile)
25th percentile (lower quartile)
Minimum observation
Interquartile Range (IQR)

Figure 1: Box Plot Definition

SUMMARY PLOTS

The box plots in figure 2 below were generated using all points tested that did not involve printing, separated by host. This graph gives a rough indication of how the host layer of the SAS MultiVendor Architecture™ is performing. Since all Version 8 data points are scaled by Version 6 times, a value of 1.0 represents equivalence with Version 6. From these plots we conclude:

1. The *median* time values for most of the hosts are below Version 6 times.
2. The *interquartile range* for all hosts is fairly tight so that we can expect more that half of the times measured to be at or below Version 6 times.
3. The *mean* values fall above the *medians* in all cases and actually fall outside the box in one case because of extreme outliers.

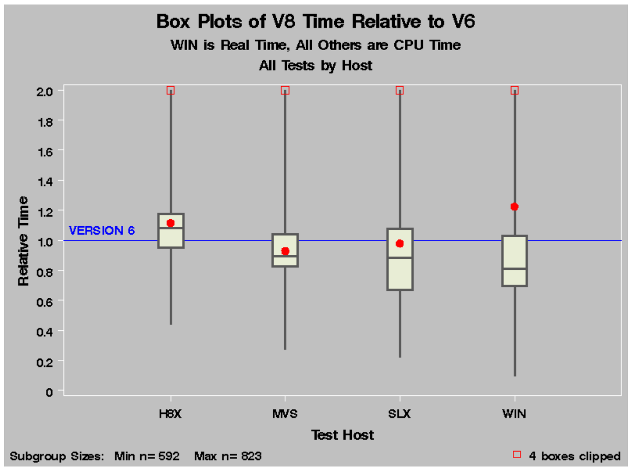

Figure 2: Overall Performance by Host

If we look at the performance of all hosts together separated by product, we get an indication of how the *portable layer* of Base SAS Version 8 is performing. Again, we see encouraging results that indicate, for the areas tested, Version 8 has performance comparable to Version 6. Note that some of the outliers identified by this study have already been corrected in Version 8 TS1. Others will be addressed in Version 8 TS2.

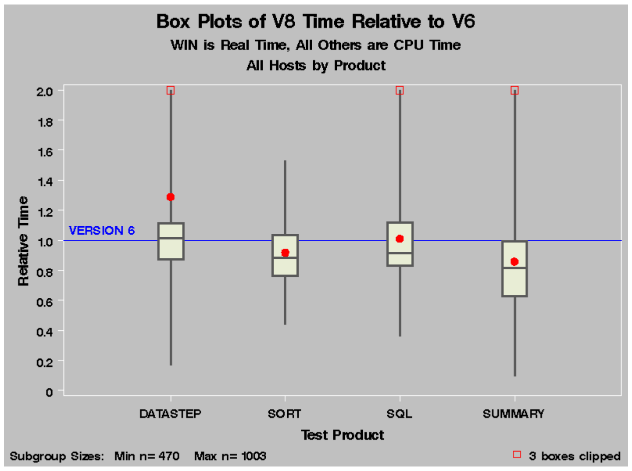

Figure 3: Overall Performance by Product

PRODUCT RESULTS

In the sections that follow, selections from the detailed results of each product area will be presented. Unfortunately, space restrictions prevent us from presenting all the test results.

DATA STEP

The graph in figure 4 below gives a summary of the DATA Step performance broken-out by test host. As can be seen, the median times are close to or below Version 6 times for all hosts. However, extreme ourliers cause the mean values to fall on or above the upper quartile in all cases. Some of these outliers have already been corrected for Version 8 TS1. Others are under investigation.

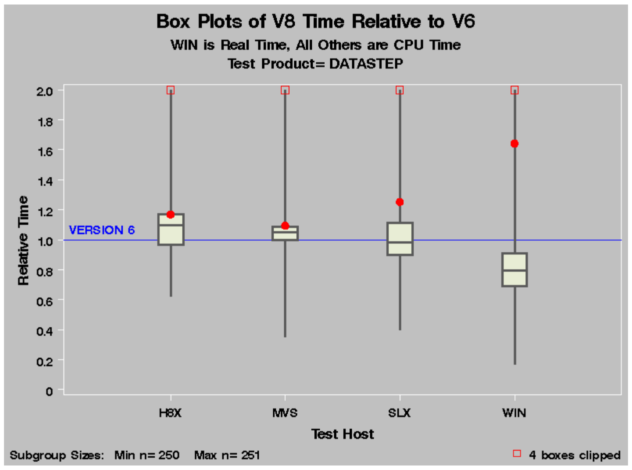

Figure 4: DATA Step Performance by Host

Since the DATA Step can be used in so many diverse ways, it would be difficult to arrive at representative DATA Step test examples. Instead, a series of *micro tests* were created that each evaluated small elements of the total scope of functionality. The elements evaluated included *input* and *put* statements with various formats and informats. Figures 5 and 6 below are examples of detailed test results. Version 8 and 7 relative times for the particular test are shown as horizontal bars while Version 6 reference time is represented by the vertical line at the 100% level.

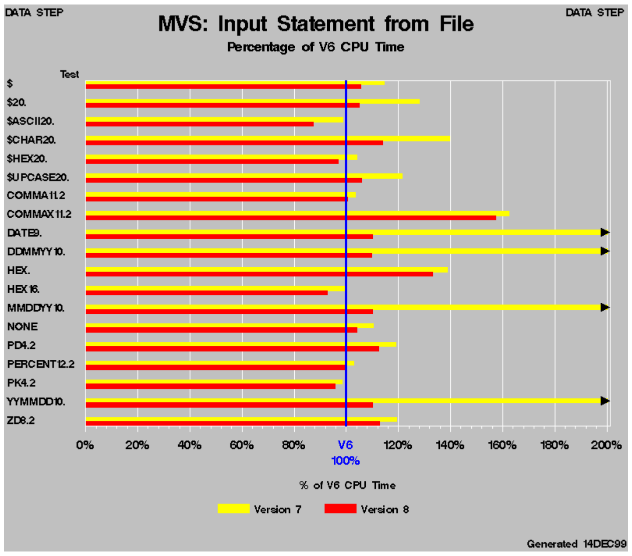

Figure 5: DATA Step Input Statement on MVS

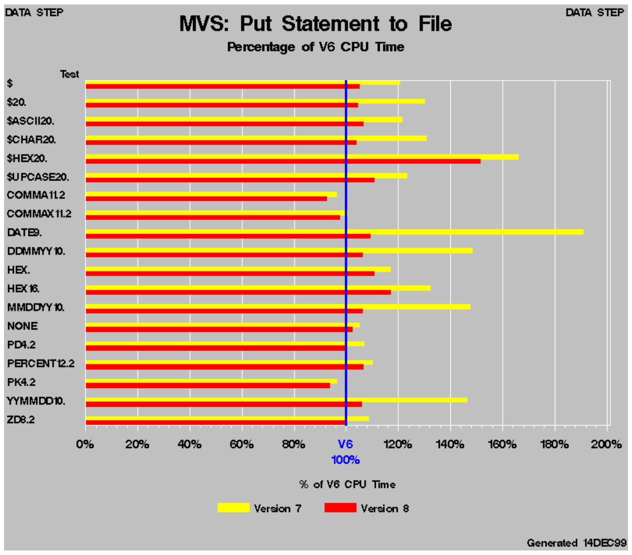

Figure 6: DATA Step Put Statement on MVS

GENERAL I/O

To test general I/O characteristics of the system, we used the DATA Step *SET* statement. An example of these test results is shown below in figure 7. The summary box plot in figure 7 below gives the results of these tests which includes simple sequential data set loads, SETs with BY conditions, and SETs with POINT and random KEY operations.

These plots show that three of the four hosts have mean times that are within 20% of Version 6 times. Windows NT is a notable exception due primarily to extreme outliers. These outliers are in the area of random access as is shown in the detailed graph, figure 8, for SET operations. While sequential access of large data sets has improved significantly, random access performance has declined. The degradation is currently under investigation.

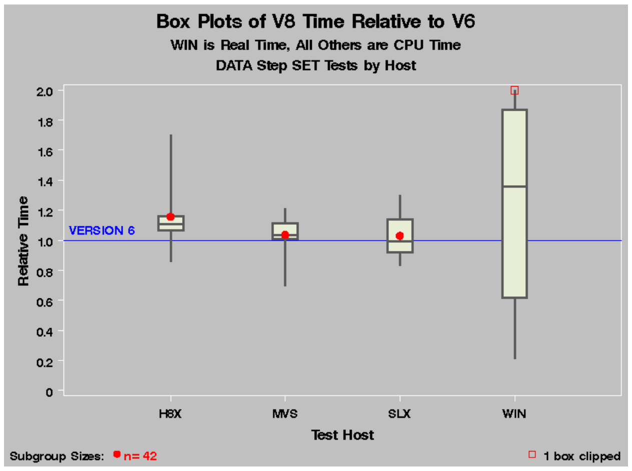

Figure 7: DATA Step SET Tests by Host

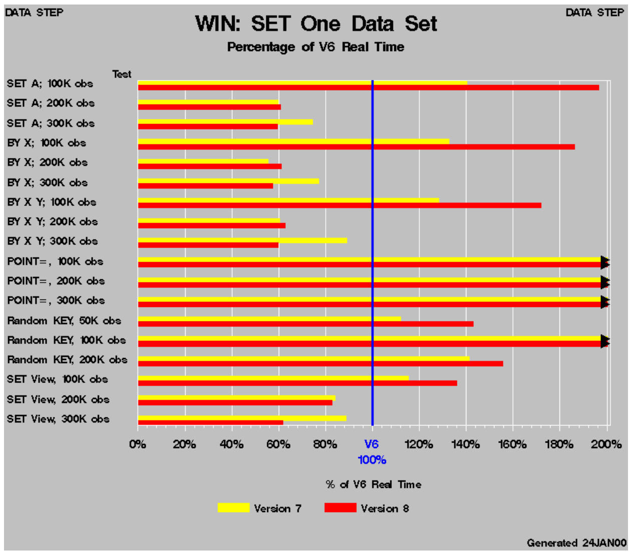

Figure 8: DATA Step SET Tests for Windows NT

PROC SORT

The overall test summary for PROC SORT is shown below in figure 9. The plots show that the performance of the portable sorting code has improved for three of the four hosts tested.

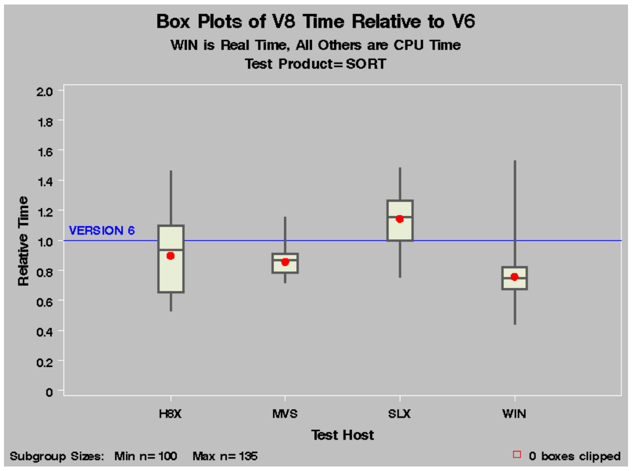

Figure 9: PROC SORT Overall Performance by Host

The data sets sorted range in size from 10,000 to 400,000 observations. Sort keys ranged from one to sixteen eight-byte numeric variables where key differences were always isolated to the last element to force complete key comparison. In each case, the data sets sorted contained only the key (BY) variables. This was done to emphasize differences in the sorting algorithms by minimizing I/O overhead for a given key size. Memory control options such as MEMSIZE and SORTSIZE were adjusted such that each test was completed in-memory without opening a temporary utility file. Again, this was done to emphasize the sorting algorithm rather than disk I/O.

It is often the case that data being sorted has some initial ordering. For example, a previously sorted data set may have new records appended to its end before resorting. For this reason, the test data sets had four initial orderings: random, pre-sorted, reverse-sorted, pre-sorted runs and reverse-sorted runs, to evaluate portable sorting performance in these common cases. Version 8 portable SAS SORT has undergone algorithmic adjustments that make sorted partially sorted data as much as 40% faster as shown in the detailed sorting graphs below.

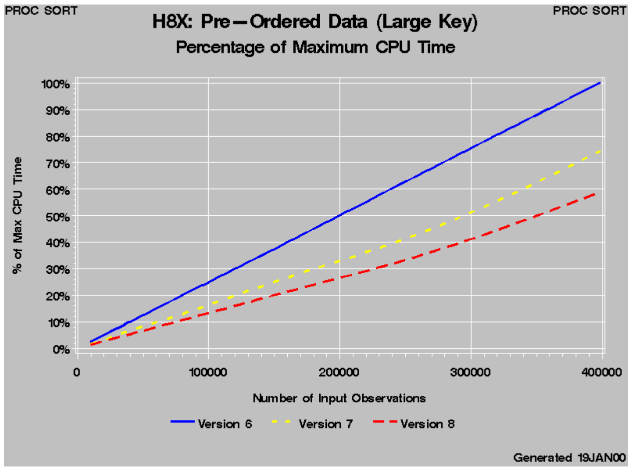

Figure 10: Sorting Pre-Ordered Data on HP-UX

More modest gains are found on other hosts as is illustrated by the MVS plot below in figure 11.

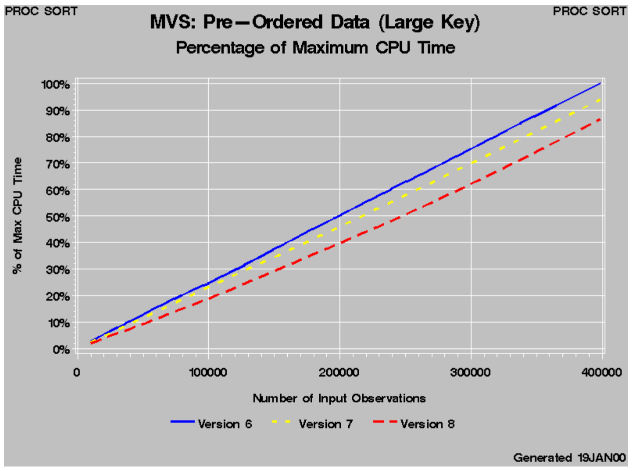

Figure 11: Sorting Pre-Ordered Data on OS/390

Conversely, the sharpest performance decline was seen on Solaris for reverse-ordered data of medium key length as shown in figure 12 below.

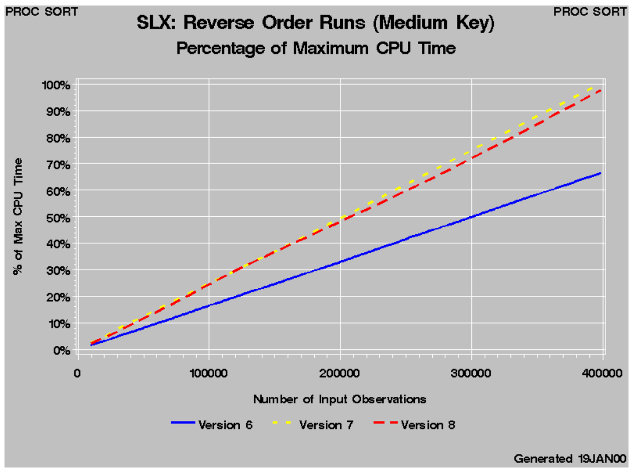

Figure 12: Sorting Reverse-Ordered data on Solaris

Naturally, this performance degradation is being investigated.

PROC SQL

To evaluate the portable SQL procedure code, the most CPU intensive operations were selected. These include four techniques of inner and outer joins and the except, intercept and union operators. Each test was performed over a range or input table sizes. The tests generate output data sets only. No tables were printed. Overall results for PROC SQL are shown below in figure 13.

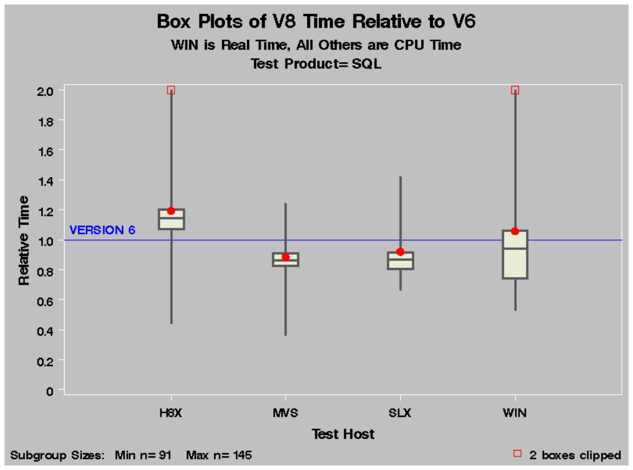

Figure 13: PROC SQL Overall Performance by Host

For the most part, PROC SQL Version 8 times fall below the equivalent Version 6 times. In most cases, significant improvements have been made between Version 7 and Version 8 in order to restore V6 or better performance. The graphs below show the gains made on Solaris for inner joins. In some cases, Version 6 performance is exceeded, in others such as joins using indexes, Version 8 performance does not yet match that of Version 6. Index access performance is currently under investigation. .

Another area of performance improvement for PROC SQL is the addition of *Implicit PassThrough* for version 7. This allows a query to be passed to an underlying SAS/Access® data provider transparently whenever possible to minimize data transfer. Evaluating the performance advantages of Implicit PassThrough is beyond the scope of this paper.

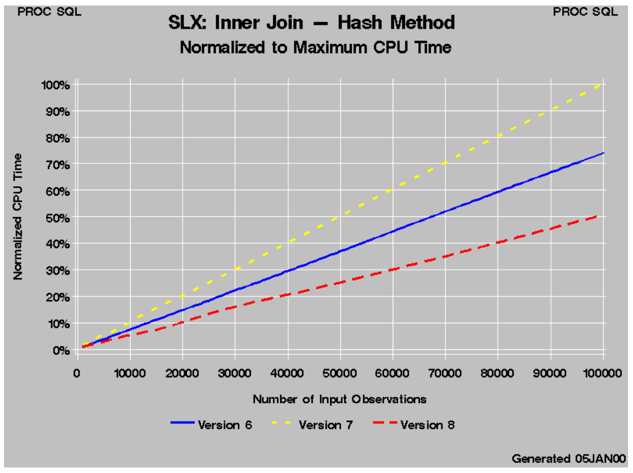

Figure 14: PROC SQL Hash Method Inner Join on Solaris

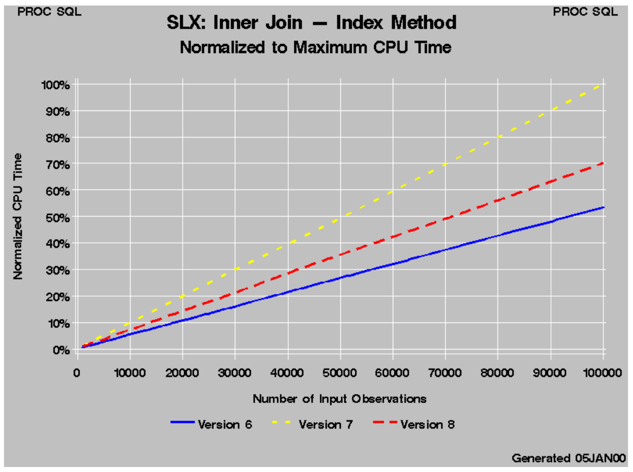

Figure 15: PROC SQL Index Method Join on Solaris

PROC SUMMARY

The test cases for PROC SUMMARY/MEANS evaluated only output data set generation. The tests fell into two broad areas, tests which used the CLASS statement to subdivide input data and those that did not. The results of these tests can be seen in figure 16 below.

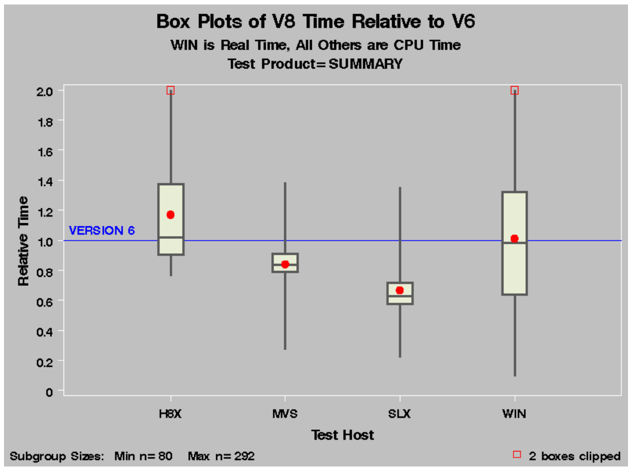

Figure 16: PROC SUMMARY Overall Performance by Host

For tests using the CLASS statement, input data sets with a constant size of 250,000 observations were generated repeatedly, in which the variation in four input class values (*cardinality)* increased so that the range of output observations for the NWAY type varied from 1 to over 120,000. For the classification studies, three levels of statistics were generated for eight analysis variables.

The first level (zero) recorded frequencies only. The second level (level 1) generated simple statistics that did not require the corrected sum of squares to be computed such as *min*, *max* and *mean*. The third level (level 2) required the corrected sum of squares to compute *variance* and *standard deviation*. A fourth level collected only ID variables. For all these tests, system memory options such as MEMSIZE were set such that the jobs completed in-memory. Low memory execution was evaluated in a separate set of tests.

In most cases, classification performance of Version 8 was significantly better than Version 7 and equal to or better than that of Version 6. The series of graphs below show PROC SUMMARY performance on OS/390.

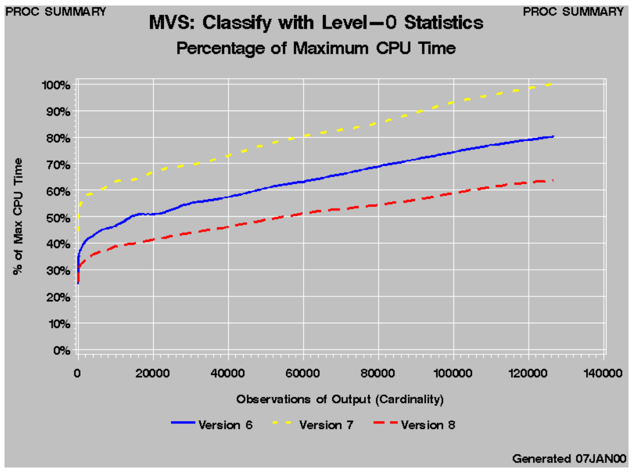

Figure 17: PROC SUMMARY Classification Level-0 OS/390

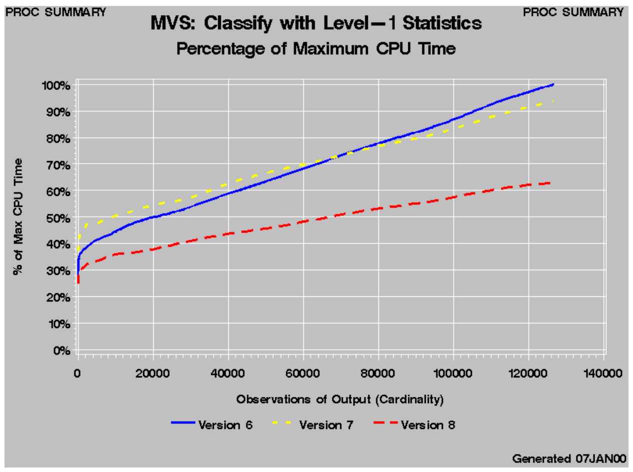

Figure 18: PROC SUMMARY Classification Level-1 OS/390

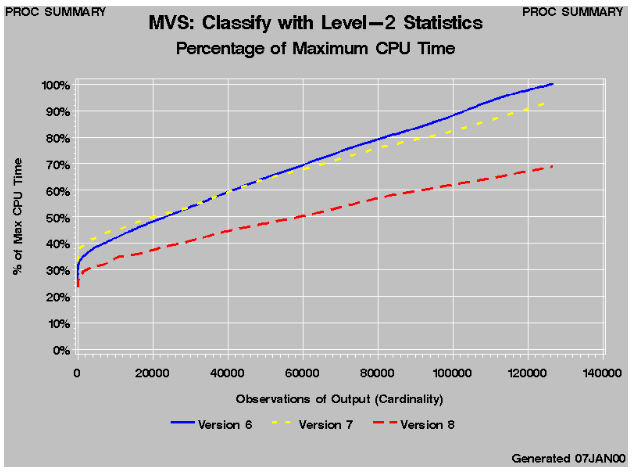

Figure 19: PROC SUMMARY Classification Level-2 OS/390

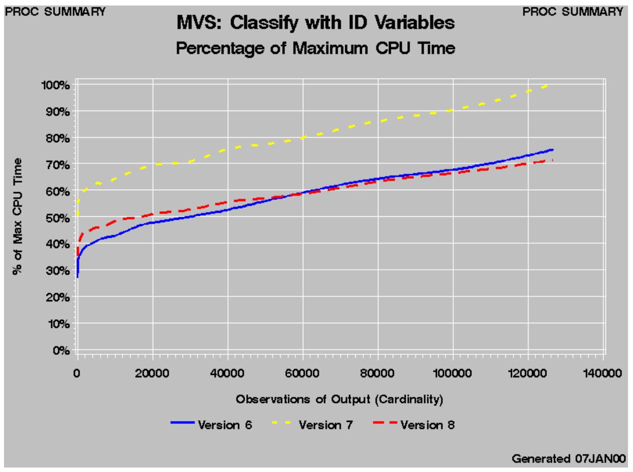

Figure 20: PROC SUMMARY Classification with ID OS/390

Although in most cases, Version 8 performance is better that Version 6, ID variable collection in Version 8 SUMMARY still needs refinement. The decline in ID collection speed is most likely due to the increased complexity of the code to accommodate the new Top-N capabilities of PROC SUMMARY/MEANS.

For Version 7, the PROC SUMMARY/MEANS internals were extracted into a library so that other Base reporting procedures such as PROC TABULATE and the REPORT procedure could share them. At that time, low memory performance was enhanced. Below is a graph showing the results of the classification test with level-2 statistics being run under low memory conditions. The sharp upward turn in the Version 6 line indicates the point when PROC SUMMARY opens a temporary utility file due to the cardinality of the problem. The downward offset of the Version 8 line from the Version 7 line is the result of work on the in-memory program overhead, which affects both in-memory and low-memory execution.

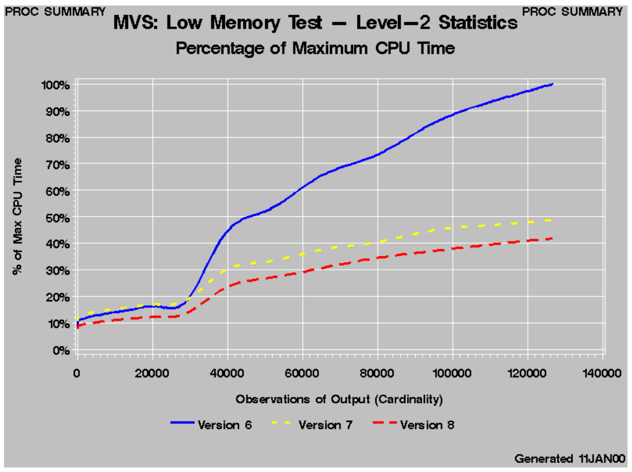

Figure 21: PROC SUMMARY Low-Memory Test on OS/390

Non-classification tests of PROC SUMMARY were designed to gather level-1 and level-2 statistics on from 10 to 1000 analysis variables for data sets ranging from 100 to 100,000 observations in length. In most cases, the startup overhead of the new, more complex SUMMARY code caused the smaller cases to run more slowly than the Version 6. Larger cases, however, ran more quickly as can be seen in figure 22 and 23 below.

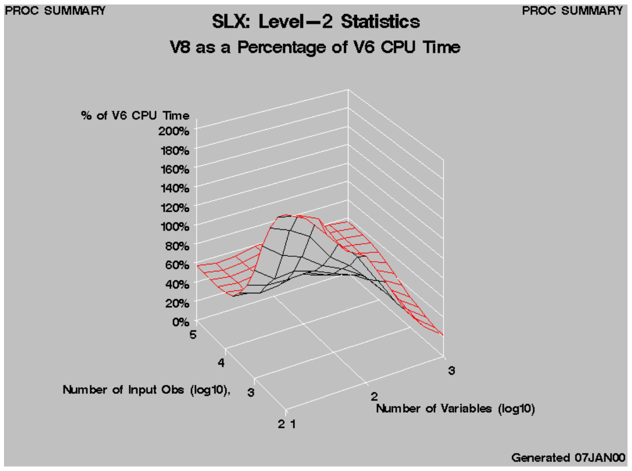

Figure 22: PROC SUMMARY Level-2 Test on Solaris

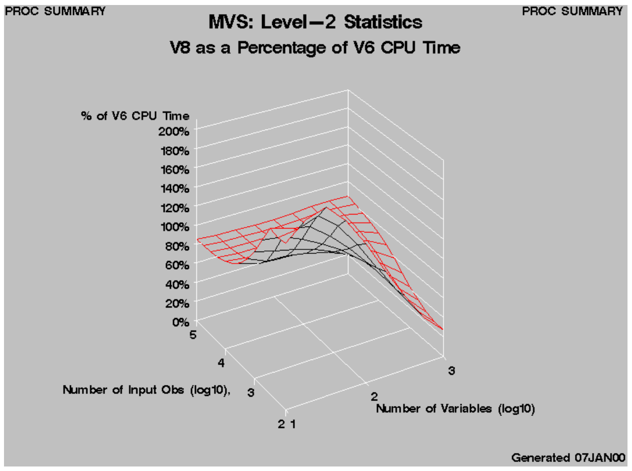

Figure 23: PROC SUMMARY Level-2 Test on OS/390

This same test on Windows NT showed degradation at the extreme range of data set size, which is the back corner of the graph in figure 24. This problem is currently under investigation. One positive note is that startup processing large numbers of analysis variables has improved for all cases.

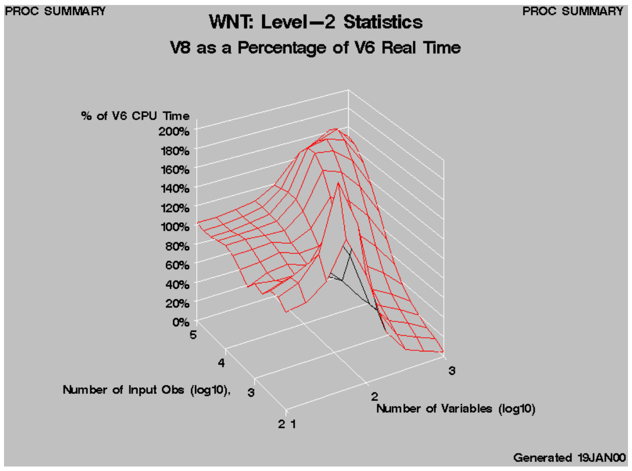

Figure 24: PROC SUMMARY Level-2 Test on Windows NT

For Version 7, PROC SUMMARY added new performance options to both the procedure statement and the CLASS statement. The *NOTRAP* procedure statement option improves the speed of generating statistics by short-circuiting the code used to recover from floating-point exceptions. NOTRAP can be used when you are confident that the data being summarized is not likely to cause overflows. The CLASS statement has the new *GROUPINTERNAL* option, which causes classification variables to be grouped by their internal values rather than by their formatted values. GROUPINTERNAL can improve performance when the formatting process does not make the class variable values less unique than their internal values and therefore has no affect which observations are grouped together by classification.

PRINTING

Although this study focused on non-printing applications, some print testing was done. For Version 7 and beyond, all printing in the SAS MVA™ product is channeled through the new Output Delivery System (ODS). ODS provides a great deal of new flexibility for creating output in different forms such as HTML. Along with this flexibility comes some unavoidable computational overhead. The plot below shows the relative Version 8 time for DATA Step examples that exercise the ODS system via the PUT statement to produce standard monospace output to the SAS Listing. As can be seen, there is an average slowdown of approximately fifty- percent for three of the four hosts.

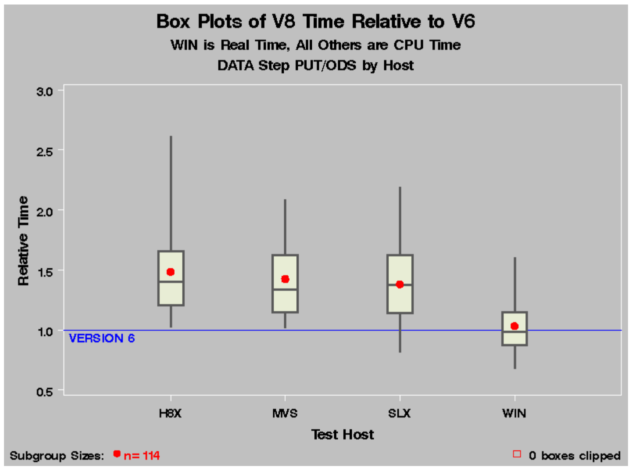

Figure 25: DATA Step PUT Statement via ODS by Host

However, not all printing applications have suffered a slowdown. Figure 26 below shows a comparison of print times for a series of large TABULATE procedure runs which use ODS to process its output. Using the same basic data set that was used for the PROC SUMMARY classification tests presented above, a simple PROC TABULATE statement was executed that analyzed 250,000 observations each with four CLASS variables and eight analysis variables. As with SUMMARY, the cardinality of the class data was expanded with each successive run so that PROC TABULATE printed from 1 to 10,000 pages of output. Figure 26 shows that Version 8 TABULATE prints monospace output via ODS roughly forty-percent faster than Version 6 for large jobs. Readers should also note that Version 8 PROC TABULATE no longer has the 32k levels-per-dimension restriction. This limitation prevented Version 6 TABULATE from completing the test series.

When alternate output destinations are used such as HTML additional overhead will be encountered. Not only is the HTML output more complex to generate than monospace output, it also results in a great deal *more* output because it includes not only the printed characters but also the markup language that surrounds them.

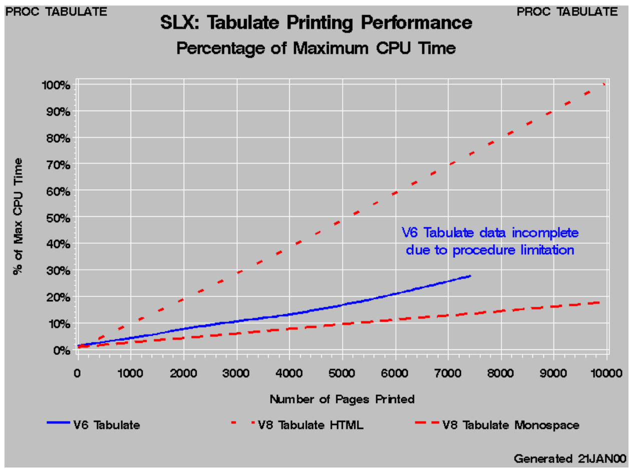

Figure 26: PROC TABULATE ODS Printing on Solaris

CONCLUSION

Our testing has demonstrated Version 8 Base SAS is comparable in performance to Version 6 with regard to execution speed. However, the exact performance that each user will experience is highly dependent on their mix of computing tasks. For the most part, large data intensive operations should perform as well as Version 6 on average. Some tasks such as sorting and summarizing may even run noticeably faster with Version 8. However, smaller, print intensive tasks may run slightly slower in Version 8 due to increased setup time for the ODS system.

Performance analysis and improvement is an ongoing part of our development effort. We expect to meet or exceed Version 6 benchmarks for all critical data processing operations during the Version 8 life cycle.

ACKNOWLEDGMENTS

The following Base SAS R&D Developers and Testers contributed the information presented in this paper:

Douglas Christie
Evan Dean
Scott Mebust
Jason Secosky

Additional consultation was provided by William Heffner and Lewis Church Jr.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Robert Ray
SAS Institute Inc
SAS Campus Drive
Cary, NC 27513
Phone: (919) 677-8000 ext. 6605
Fax: (919) 677-4444
Email: Robert.Ray@sas.com
Web: http://www.sas.com/rnd/base/