Paper 282-28

An Inside Look at Version 9 and Release 9.1 Threaded Base SAS® procedures

Robert Ray, SAS Institute Inc. Cary NC

ABSTRACT

This paper looks at the changes in key SAS® Base procedures which now incorporate threading to achieve significant performance improvements on SMP architectures. The paper is relevant to the following SAS procedures: SORT, SUMMARY, MEANS, TABULATE, REPORT and SQL. We look at software scalability in general and specifically how threading has been utilized in these procedures to enhance scalability. Sample test plots illustrate how threading can reduce the real-time to completion. We also cover the new system and procedure options that pertain to threading such as THREADS and CPUCOUNT. In addition, we cover guidelines regarding utility I/O configuration that will help optimize scalability and discuss the new startup parameter UTILLOC.

INTRODUCTION

Over the past decade, the volume of data that organizations collect and analyze has skyrocketed. During this same time, hardware vendors have stepped up to the plate with ever increasing CPU speeds, multi-processor architectures and, higher I/O rates. With the introduction of the SAS® Scalable Architecture (SSA) in Version 9, SAS developers are, for the first time, able to access the power of multi-processor architectures features from the portable MVA (Multi-Vendor Architecture) SAS. As a result, we have been able to introduce limited multithreaded capabilities into many of the mainstream Base SAS procedures such as SORT, SUMMARY/MEANS, TABULATE, REPORT and SQL. The SSA now allows a single SAS session to leverage multiple I/O channels and multiple CPUs to drastically cut time-to-solution for large jobs.

In the past, MVA SAS was generally regarded as "I/O bound" when the internal computation was hindered by not being able to receive data fast enough. This is often not the case anymore. Today's high performance I/O systems can deliver data at CPU-gagging rates of hundreds of megabytes per second. With these rates, it's easy for a SAS session to become CPU bound, limiting the data input rate and extending the time to solution. The multithreading made possible by the SSA now allows input data to be processed in parallel to better keep pace with the new high speed I/O devices. This new ability for selected SAS base procedures to respond to enhanced hardware is the essence of *software scalability*. The next sections will look at the various areas that contribute to the enhanced scalability of Version 9.

SOFTWARE SCALABILITY BASICS

Software scalability refers to the software's ability to take advantage of hardware improvements that multiply the number of functional units available within a single process space. These functional units include both CPUs and I/O devices (disks and controllers).. The ability to have multiple independent threads executing on separate CPUs that communicate directly via main memory is central to a software's ability to scale. The SAS SSA is specifically designed to give application writers the tools necessary to leverage symmetric multiprocessor (SMP) architectures which allow multiple process threads to execute in parallel on separate CPUs while sharing a common memory space

Scaling I/O involves combining separate disk drives either with hardware RAID or software partitioning or both to multiply the effective transfer rate of a single file by spreading it parts over multiple disk spindles and controllers. These connected drives work as one to speed data into main memory. The new SAS SPDE multi-partition engine [1,2] provides this ability to divide I/O operations across multiple files systems and I/O controllers in order to multiply the effective data transfer rate into the application code. With the SPDE engine, it is also possible for data subsetting operations such as where clause evaluations to be performed in parallel on separate data partitions. In addition, the new SPDE engine supports sophisticated "bit" indexing schemes which further enhance the data query speed. The access engines have also been enhanced to take advantage of multithreaded parallel access where possible to ensure that data transfer rate into the SAS System from other data stores is maximized.

THE V9 FOCUS

Base SAS development has focused on cutting the *real-time* to solution for core tasks such as sorting or summarizing large data sets. When trying to speed a single task or problem, there are two types of scalability to consider - the inherent scalability of the problem in question and the scalability of the software solution for that problem. Problem scalability can vary greatly. The problem of sorting, for example, generally scales computationally on the order of $Nlog_2N$, where N is the number of records to be sorted. However, if the I/O device cannot keep pace with the CPU, then the scalability will be linear with the size of the file (N). A full SQL join will scale computationally as $N*M$, where N and M are the table row counts. Therefore, doubling the number of observations for both tables in a full join could consume four times the CPU resources. However, if the

process were I/O bound, the actual scalability would be linear with the combined size of the tables (*N*+*M*). Therefore, reducing time-to-solution is a complex problem involving both CPU and I/O optimizations.

With software scalability, the goal is to apply additional physical resources (CPUs or I/O channels) and have the real time-to-solution be lowered by a proportional amount. The real time is the focus here, and not the combined CPU time. Some extra CPU cycles are consumed to manage a set of process threads across multiple CPUs but by keeping the granularity of thread interaction fairly coarse, this additional CPU time is negligible.

The portion of the original problem that can actually be processed in parallel governs the amount of scalability achieved from the software solution. For instance, although PROC SORT can read portions of a partitioned data set in parallel, the output data set must be written in a linear fashion to preserve the sorted state. If writing the data takes 50% of the original time, then only 50% of the process is scalable; this represents the limit of scalability for this problem. Further improvements in time-to-solution can only be achieved by increasing the speed of the output I/O devices.

For the most part, the threaded statistical procedures (e.g. PROC REG) are predominantly CPU bound. They can show perfect scalability with regard to the CPUs applied to the problem. Base procedures such as SORT and SUMMARY, on the other hand, can easily be either CPU or I/O bound or both within different phases of a single execution. And so, the focus has been on reducing the real time of a task to the sum of the I/O times involved. The goal for the threaded base procedures has been to make the real time for the task equal to the time it takes to read in the input data and write the output results. Of course, it stands to reason that the I/O rate and CPU power need to be in balance for optimum performance. Coupling a slow I/O subsystem to a multiprocessor machine will give little opportunity to reduce processing time. Whether or not a SORT or SUMMARY is CPU bound, and therefore a candidate for scalable threading, is to a large part dependent of the transfer rate of the input I/O device. For the V9 development effort, an input device was considered as “high speed” if it’s transfer rate was 50 Mbytes/sec or more.

PARALLEL COMPUTATION MODELS

When a problem is CPU bound, carefully applied parallel computation using modern thread constructs will result in lowered real times. Three common models of parallel computation are the *boss-worker* model, the *pipeline* model and the *peer* model. All these models allow a computationally intensive task to be divided and distributed to multiple CPUs within a shared memory model using lightweight process threads. The new threaded Base SAS procedures each use one or more of these techniques, along with parallel I/O where appropriate, in order to reach its scalability limit. The two most commonly used models are the pipeline and peer. The pipeline model employs a series of threads that each execute a *unique* subset of a given task on a single block of data and then pass the partial results to the next stage for further computation. The throughput of a pipeline is limited by its slowest stage and the scalability is limited by the number of stages that can be defined but the problem space is not divided so there is no required merging of partial results. The peer model uses multiple threads to execute the *same* task on different segments of data in parallel. This model scales symmetrically with the number of CPUs but it produces multiple partial results sets that much be merged together at the end of input processing.

In order to utilize threads, the traditional SAS MVA procedure will create threaded code segments that essentially run *outside* of the MVA environment synchronizing back once some portion of work is complete. Since these new external segments do not execute in the current SAS MVA thread, they do not detract from and may even enhance the interactivity of the SAS environment during the execution of long-running procedures on SMP machinery. When reading data in multiple parallel partitions from the new SPDE data source, both I/O and computation occur outside the MVA thread which just waits for processing to complete. However, when reading data from any of the traditional data sources such as the Base or Access engines, a common model is to have the traditional SAS MVA thread do nothing more that gather observations into large buffers and pass them across into the threaded space for processing. Once processed, the buffers are retuned to the MVA thread to be reused. This block-mode I/O replicates the conventions of the new SPDE parallel threaded data reads where the procedures are presented with blocks of records rather than a single record at a time.

UTILITY I/O

Not only have we been considering how to use multiple CPUs to keep pace with input data rates, we have also closely evaluated how our threaded procedures interact with utility files. We have re-worked algorithms to avoid frequent disk head seeks. For example, our new threaded sort employs a sophisticated new predictive read ahead model that allows sort utility files to be read back at nearly the same rate as a straight sequential read. In addition, the threaded Base procedures leverage the new ability to define multiple utility file paths, described below, to reduce read/write contention on a single device. For instance, if a utility file requires sorting, the sorting service can now access utility space that is on a separate file system from that of the utility file being sorted. In this way, the reading of the utility file being sorted does not conflict with the writing of any temporary sort files and likewise, the reading of the sort utility files will not

contend with the re-writing of the target utility file. Both the SUMMARY and SQL procedures take advantage of this scheme.

MULTIPLE UTILITY FILE PATHS

With SAS Version 9, a new startup option is available, UTILLOC. The UTILLOC option allows the user to specify one *or more* utility file paths to be used by SAS threaded procedures. This extends rather than replaces SASWORK. Using UTILLOC to specify multiple utility file paths has the potential to enhance the throughput of the new threaded procedures that have been designed to use this feature provided that the file paths exist on separate I/O devices. If UTILLOC is not specified at startup, it will take on the same value as SASWORK.

To best leverage multiple utility file paths, the SUMMARY, TABULATE and REPORT procedures currently use the first UTILLOC path for scratch space while the shared threaded sorting utility defaults to the second location. Other procedures that use the threaded sorting utility such as the SQL procedure still use the SASWORK directory for their utility work but may also call the threaded sort utility for order-by and group-by operations. In view of this, it is recommended that for optimal performance with only two utility paths, SASWORK and UTILLOC path #1 be the same path while UTILLOC path #2 point to a separate I/O device. For example:

```
-work "d:" -utilloc "('d:' 'e:')"
```

See your host guide for specific instructions on setting UTILLOC on your host.

NEW SYSTEM OPTIONS FOR THREADS

Two new global SAS options have been provided to help tune and control threaded procedures, CPUCOUNT and THREADS. These two simple options are used to influence but not necessarily control the behavior of the new threaded procedures. Each procedure is free to interpret the settings of these options as appropriate for the task. The exception to this rule is the negative setting for the THREADS option, NOTHREADS. In the absence of a local override, each procedure will revert to a non-threaded behavior when THREADS=NO.

The THREADS option is a simple switch to turn procedure threading on and off. By default, THREADS is set *on* in SAS Version 9. This signals the procedures that have been modified to use threads that threading is *allowed*. The procedures may include additional heuristic logic to determine if using multiple threads would be beneficial for a given case. For example, the SUMMARY procedure will not use threads by default if a BY statement is present since there is potential for performance degradation when there are a large number of very small BY-groups. However, the SUMMARY procedure also includes a new THREADS procedure statement option which can be used to force threading on regardless of the value of the global THREADS option or the presence of a BY statement. Setting the global THREADS option to NOTHREADS will cause the threaded procedures to revert to their non-threaded behavior. Likewise the NOTHREADS command can be used directly on the procedure statement.

CPUCOUNT is intended as a *guide* to the procedure on how to setup its thread model. It *does not* limit the number of threads created by the application or the number of CPUs that might be loaded by the SAS session. By default, CPUCOUNT is set to ACTUAL which means that the SAS system will determine how many CPUs are available to its process space and set CPUCOUNT to that value. CPUCOUNT may be set artificially high or low to alter the behavior of the threaded procedures but may always be returned to ACTUAL to have the SAS system re-query the operation system for the current number of CPUs available.. If the CPUCOUNT is set to one, the procedure my elect not to use auxiliary threads at all even if the global THREADS option is set to YES

The actual number of threads that a procedure will start depends on many factors including CPU count. Some problems dictate that a certain number of processing or pipeline stages be established even though there might not be a separate CPU for each stage. Or, if CPU count is set lower than the actual number of installed CPUs and is also lower than the minimum number of separate stages in the application model, it is possible for more CPUs to be involved in data processing than the current CPUCOUNT. In order to actually limit the number of CPUs that a single SAS session may load, you will need to use an operating system provided utility, such as the *psrset* command on Sun Solaris®.

THREADED SORTING

Starting with Version 9, Base SAS will be shipped with a new high performance threaded portable sorting utility. This sorting utility will be used by the SORT procedure when THREADS=YES and portable SAS sorting is selected either by system heuristics or by explicitly setting SORTPGM=SAS. This sorting system employs several new algorithms that allow it to improve throughput by as much as eight-times over the previous portable sort given sufficient I/O. For a given I/O capability, the greater the key size relative to the record size, the greater the potential for productive parallel processing.

The threaded sort creates a sorting network which contains both peer and pipeline elements where in-memory sorting and I/O overlap efficiently. In general, sorting breaks down into internal (memory resident) and

external (utility file based) elements. The new sort improves on both of these areas. To accelerate internal sorting, multiple sorting threads accept blocks of input from one or more data partitions in parallel. Compact sort key blocks are formed to minimize CPU cache thrashing during block sorting. Sorted blocks are processed by a merging stage that employs multiple threads to overcome main-memory access latency when gathering records from the vast memory spaces that are available on today's 64bit hardware.

Below is a graph that illustrates the potential performance advantage of the new threaded SORT procedure. Real execution time is plotted for SAS Version 8.2 SORT and Version 9.1 threaded SORT over a range of data set sizes. The sort key is one 8-byte numeric out of a 64-byte record. In general, the larger the percentage of the sort record is included in the sort key, the greater the possibility for performance improvement with the threaded sort since it is operations on the key that consume a large portion of the sorting CPU time.

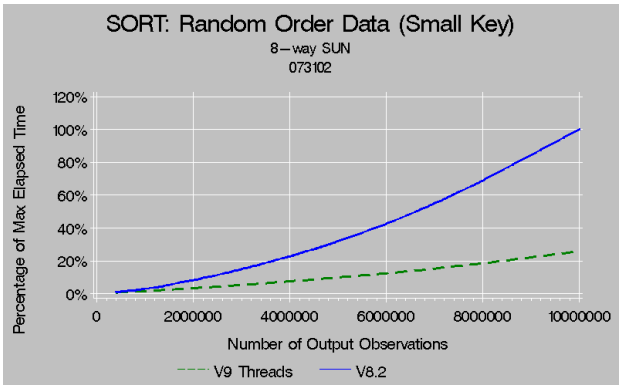


Figure 1: Internal Threaded Sort

In the area of external sorting, much work has gone into efficient interaction with the I/O devices that hold the utility file. First, to reduce the total utility storage required and therefore the total bytes that must be written and read back, the threaded sort no longer stores the sort keys in the utility file. These keys are regenerated, as needed, in parallel with the file merging thread. The result is that temporary utility file space is never greater than the input file being sorted. This differs from the non-threaded portable sort where the utility file requirements could be as much as twice the input file size.

More important than total file size however, is the disk access pattern. Fine grain random access of a file can drastically reduce the effective read rate for a sort utility file. The new threaded portable sorting service uses a unique new *disk head seek reduction* technology which allows a single utility file containing multiple concatenated sorted runs to be accessed for merging at almost the same speed as if the file were completely sorted. A dedicated thread performs predictive reads and key builds on the utility file to ensure that there is no delay in the merging process over the time required to do a straight file copy. The result is a smooth fast and quiet external sort.

NEW PROC SORT CONTROLS

As with other threaded procedures, the SORT procedure now includes a THREADS/NOTHREADS local override. To be sure you're actually using the SAS portable threaded sort, set the global SAS option MSGLEVEL to 'I'. Additionally, there is a new SAS global option SORTEQUAL/NOSORTEQUALS that allows you to globally disengage the stable sorting logic (EQUALS) which is on by default in the SORT procedure. Stable sorting means the records with equal keys will be output in the same relative order as they appeared in the input data set. The behavior is rarely needed and costs both memory and CPU time. SORTEQUALS is the shipped default to maintain backward compatibility but NOSORTEQUALS is recommended.

Memory usage for sorting can still be controlled by SORTSIZE but on many hosts, the default setting for SORTSIZE is now MAX. In this case the global SAS option REALMEMSIZE will govern the upper memory limit for internal sorting. This is consistent with the initiative to consolidate memory controls into MEMSIZE and REALMEMSIZE and away from procedure specific options such as SORTSIZE and SUMSIZE. REALMEMSIZE designates the amount of real (non-virtual) memory that is expected to be available to the SAS process. On some hosts, this value can be determined by the operating system from the hardware. Other hosts, this value can not be determined and should be set by the user.

Two new options are being developed in the SORT procedure for Version 9 time frame, DUPOUT= and OVERWRITE. DUPOUT= allows you to specify a secondary output data set that will contain observations discarded by the duplicate removal options, NODUPREC and NODUPKEY. OVERWRITE will allow SORT to delete the input data set before writing the sorted output data set. This will cut the disk space requirements for sorting by as much as 50%. Internal sorts will require no additional disk space while external sorts will need only enough space in the work directory to copy the input file. No additional space will be required in the source directory. The downside of this option is that the input data set could be lost if a failure happened during the final write phase. However, since you are essentially writing the sorted file over the original disk space, there is less likelihood that the most common write error, disk-full, would occur.

THREADED BASE REPORTING

The primary Base SAS reporting procedures, SUMMARY/MEANS, TABULATE and REPORT all have threaded capabilities in SAS Version 9.1. All these procedures share a common summary service which manages data classification and statistics generation. Because these procedures have several different modes of operations, with or without a CLASS statement, with or without a BY statement, with or without analysis variables, the underlying summary service has several distinct thread models, each with its own characteristics.

The simple approach to threading a summary process would be to simply divide the data into N separate partitions, summarize each in parallel, and merge the results. Unfortunately since the classification process consumes memory for each unique class variable value combination, separating the process into N separate spaces could multiply the memory requirements by N. To avoid this pitfall, the summary service divides the process into a series of stages which are pipelined together. Intermediate results from one stage are passed down to the next along with a block of data records. In this way, multiple CPUs can be involved in the summarization process without increasing memory requirements. In addition, this model nearly eliminates the contention between threads for access to shared memory. The downside of a pipeline model, where each stage has different operations, is that an imbalance can occur. One stage may require much more CPU time to complete than the next leading to bottlenecks and poor scalability. To address this issue, peer threading within a given stage is employed to restore balance to the model. Ideally, each unit of work given to a thread across the model would be roughly even. In this case, there will be no bottlenecks in the pipeline.

Without CLASS (or GROUP) variables, the threading model for the summary service is relatively simple. N peer threads aggregate statistics into N separate spaces and these spaces are merged at the end of data processing. N, in this case, is generally equal to CPUCOUNT. The model scales easily and usually has an execution real-time equal to the basic data read time. So, for example the following two steps should have approximately the same real time if a sufficient number of CPUs are available to keep pace with the I/O device:

```
data _null_; set foo;
proc means data=foo; output out=bar;
```

Below is a graph which illustrates the scalability of the SUMMARY procedure which has no CLASS statement. Perfect behavior would place the dashed line of the threaded SUMMARY directly over the dotted line of the I/O meaning that the real time to summarize the data was equal to the time to simply read the data off disk and discard it.

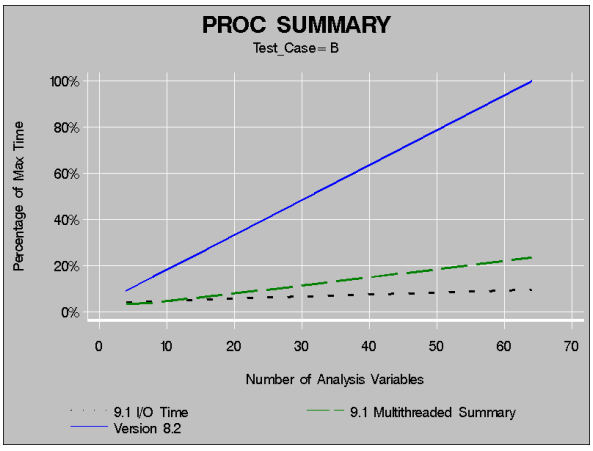


Figure 2: SUMMARY w/o CLASS, 8-CPUs

Because classification is essentially a sorting process, using a CLASS statement on any of these procedures requires that the most complex pipeline be setup. It is also most subject to pipeline imbalances. A typical classification pipeline to build the NWAY type includes four stages, one to check for missing class variable values, one multi-threaded stage to normalize class variable values, one to lookup the aggregation space from an in-memory table, and a final multi-threaded stage to update the statistics within a given aggregation space. This pipeline is fed data in blocks from one or more dedicated I/O threads. The graph below shows a comparison between SAS Version 8.2 and Version 9.1 running on an 8 CPU machine for SUMMARY procedure steps that have a constant number of analysis variables (4) but whose class variable count varies from 2 to 64. The cardinality of the data is approximately 100/1(one output observation is produced in the NWAY for each 100 input observations).

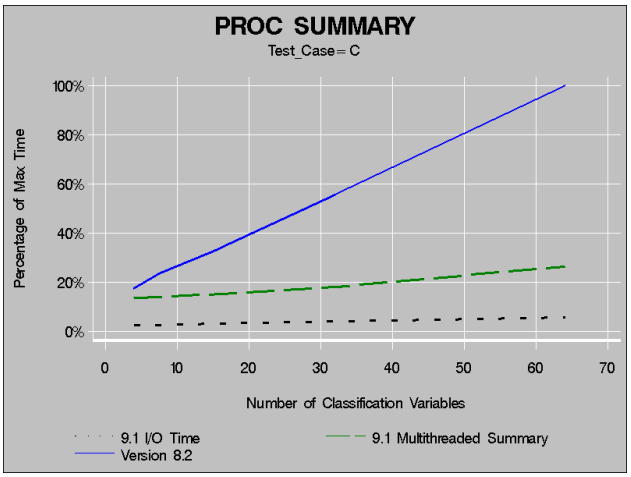


Figure 3: PROC SUMMARY w/CLASS 8-CPUs

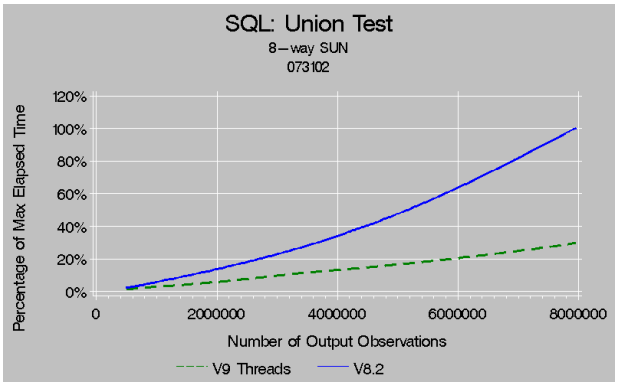
As you can see, the scalability of the problem increases as the number of classification variables increase but there is still a significant gap between the procedure time and the basic I/O time. This is the result of threading overhead and pipeline imbalances. Future development will focus on bringing the two lines together so that the classification process can better keep pace with I/O.

EXTERNAL SUMMARY PROCESSING

When memory is exhausted in the process of expanding one of the summary crossing tables, partial results are spooled to a utility file to free memory. In prior releases, the process used a *repeated union* technique which maintains only unique summary class values on disk. This keeps utility file requirements at a minimum but when the cardinality of the data is high and storage requirements greatly exceed memory, the repeated union technique becomes very inefficient. For Version 9.1, the summary system now uses a multi-way merge technique for generating the NWAY crossing on disk. This represents a classic speed for space tradeoff since the utility file can now contain duplicate class values on disk prior to the final merge.

THREADED SQL

In the past, the SQL Procedure has used the SAS portable sorting routines to perform ordering operations as needed to implement *order by, group by* and other operations such as *merge joins*. For Version 9.1, SQL will use the new threaded sort utility to perform row ordering. Because sorting time can dominate many large SQL queries, leveraging the parallel sorting technology can reduce query execution time in the same way that PROC SORT times have been reduced. This feature is automatic if the global THREADS option is set to YES. Below is a plot that shows the impact of threaded sorting on the *union* operation in SQL, one of the many operations that order rows as part of the implementation.



CONCLUSION

Version 9 represents the first steps in a bold new direction for portable application development at SAS toward being able to provide customers with software that better leverages their high performance hardware investment. Whether the hardware is a 2-CPU workstation or a 64-CPU server, Version 9 provides the potential for significant improvements to your throughput. However, in order for scalable software to succeed, you must pay close attention to hardware configuration details such as I/O layout, data partitioning and utility file placement. Be aware of the effects that the new options THREADS, CPUCOUNT, REALMEMSIZE and UTILLOC have on the behavior of the procedures. More than with any previous version of SAS, the phrase "*your mileage will vary*" applies.

REFERENCES

Doninger, C. (2002), "Up and Out: Where We're Going with Scalability in SAS® Version 9," *Proceedings of the Twenty-seventh Annual SAS Users Group International Conference*
(http://www2.sas.com/proceedings/sugi27/p279-27.pdf)

Clifford, B. (2002), "Scalable Access to SAS Data" *Presented at the Twenty-seventh Annual SAS Users Group International Conference.*
(http://www.sas.com/rnd/papers/sugi27/scalable_access.pdf)

ACKNOWLEDGMENTS

I would like to recognize the contributions of the following SAS staff members:

- Scott Mebust – *developer of the threaded sort*
- Peng Xu – *research assistant*
- Evan Dean - *development tester*

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) 531-6605
Fax: (919) 677-4444
Email: Robert.Ray@sas.com
Web: http://www.sas.com/rnd/base/

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.