Paper 279-27

Up and Out: Where We're Going with Scalability in SAS® Version 9

Cheryl Doninger, SAS Institute Inc.

ABSTRACT

This paper gives an overview of the ways that SAS is addressing performance through scalability in SAS Version 9. Scalability features have been implemented in many areas of SAS Version 9 to allow your applications to scale up and scale out. These include:

- Multi-Process (MP) CONNECT,
- the Scalable Performance Data Engine (SPDE engine),
- certain SAS/ACCESS® engines,
- several scalable SAS procedures,
- the Scalable Performance Data (SPD) Server®,
- the Open Metadata Server, and
- the Online Analytical Processing (OLAP) Server.

The purpose of this paper is to describe each of these areas at a high level and explain how they can be used independently or in conjunction to maximize performance.

INTRODUCTION

Nearly every type of business has seen an explosion in the amount of data that it must store, process, and understand. At the same time, the acceptable time frame for turning all of this data into information continues to grow shorter. The organizations that are in the best position to handle this onslaught of data are those that are able to adjust or *scale* their hardware and applications to accommodate this increased demand.

Scalability is all about reducing the time-to-solution for your critical tasks. This can be accomplished by performing two or more tasks in parallel (independent parallelization) or overlapping two or more tasks (pipeline parallelization). This requires two things: 1) that there is at least some portion(s) of your task that can be overlapped or performed in parallel and 2) that you have hardware that is capable of multiprocessing. It is important to understand that not every application lends itself to scalability and not every hardware configuration is capable of providing scalability.

To decide whether an application should be scaled it is helpful to determine if it takes "too long" to run. This may mean that the time required to run a job exceeds the batch window of time that you have available. Or it may mean that it takes "too long" for you to get the information from your application in order to make timely decisions. Next it is important to identify the pieces of the application that seem to consume the most time. Then you can determine if these portions of your task are compute intensive or if they are I/O bound. This will help you to understand how scalable a particular task may be.

Hardware that is capable of multiprocessing would include symmetric multiprocessing (SMP) machines or multiple machines on a network each containing a single processor. In addition to the number of processors, it is important to have multiple I/O channels. This is inherent to multiple machines on a network. For an SMP machine, this can be accomplished with RAID arrays that allow you to stripe or spread your data across multiple physical disks. Even for a single threaded application, this can improve I/O performance because the operating system is able to read data from multiple drives simultaneously and synchronize the result for the application. For an application that is threaded, not only can the reads be done in parallel, but threads can be used to process the data in parallel as well.

Scalability can be addressed from two directions: *scale up* and *scale out.* It is important to realize that these are not mutually exclusive choices. Hardware vendors have responded to the need for scalability by creating SMP machines that provide increased horsepower for solving large, CPU intensive problems. Scaling up, from a hardware perspective, means increasing the number of processors, disk drives, I/O channels, etc. on a single server machine. Scaling out, on the other hand, means adding more hardware, not bigger hardware. When you scale out,

the size and speed of an individual machine does not limit the total capacity of your network.

Successfully scaled performance is not obtainable by simply installing more/faster processors or more/faster I/O devices. Scalability involves making choices between investing in SMP hardware, upgrading I/O configurations, making use of networked machines, reorganizing your data, and how much you are willing to modify your application. Achieving true scalability is a balancing act involving the choice of scalable hardware along with the right software that is specifically designed to leverage it. The portion of the original problem that can actually be processed in parallel determines the amount of scalability achievable from the software solution.

Starting in Version 8 and continuing in Version 9, enhancements have been made to allow SAS to scale up - fully utilize SMP hardware, and also to allow SAS to scale out - fully utilize distributed processors. This paper introduces the areas of SAS that address performance through scalability. Characteristics are listed for which each solution will provide performance benefit. In addition, scenarios and results of combining a scale up solution with a scale out solution to achieve maximum performance gains are described .

MP CONNECT

Multi-process CONNECT (MP CONNECT) allows you to perform work in parallel and coordinate the results into your original SAS session for the purpose of reducing the total elapsed time necessary to execute a particular application. MP CONNECT also provides a convenient way to pipe data from one process or SAS procedure to another to allow overlapping execution of some SAS procedures. By dividing time-consuming tasks into multiple units of work and executing these units of work in parallel, a job can be performed in substantially less time than if each task is performed sequentially.

MP CONNECT provides a convenient syntax for spawning *n* SAS sessions to simultaneously execute *n* tasks as independent processes and coordinate the execution and results of all *n* tasks into the original SAS session. The *n* SAS sessions or processes can all execute on the same machine with each session or process running on a separate processor. Another extremely beneficial feature of MP CONNECT is the ability to spawn multiple SAS sessions to run on any number of remote machines across a network. The remote machines can have either single or multiprocessor capabilities. This is useful for either distributing multiple independent tasks to execute in parallel as well as to "divide and conquer" a large problem by breaking it up into many smaller pieces and repeatedly distributing the pieces to multiple processes until the problem is solved. In addition, MP CONNECT can execute a SAS process running one procedure which pipes it's output as input to another procedure running in another process. This execution overlap can reduce the time to solution.

MP CONNECT is part of SAS/CONNECT® and was initially available with SAS Version 8. It works at the process level creating multiple SAS sessions or processes to execute in parallel. Because the interface to MP CONNECT is syntactic, it does require that you analyze your application to identify parallel tasks and then insert the appropriate syntax to cause these tasks to execute in parallel.

SPDE ENGINE

A new engine has been developed for SAS Version 9 called the *Scalable Performance Data Engine* (*SPDE engine)*. The purpose of this engine is to speed the processing of large data sets by accessing data that has been partitioned into multiple physical files called partitions. The SPDE engine initiates multiple threads with each thread having a direct path to a partition of the data set. Each partition can then be accessed in parallel (by a separate processor) which allows the application to analyze data in parallel, as fast as the data is read from disk. This can effectively reduce any I/O bottlenecks and substantially decrease the elapsed time to process data.

Support of partitioned data is the foundation for the performance gains possible with the SPDE engine. Partitioning the data is something that must be specified with SAS syntax when a data set is created. Once data has been partitioned, there are several areas in Version 9 where gains in performance can be seen with the SPDE engine. The first is with any data step or SAS procedure that does WHERE processing. The SPDE engine provides parallel WHERE clause processing by initiating multiple threads to apply the same WHERE clause to each of the partitions in parallel. Another is a set of SAS procedures that have been modified to take advantage of the ability to read blocks of data in parallel from multiple partitions. These procedures will be covered

in more detail in the next section. Another area for potential performance gain by using partitioned data is to reduce I/O bottlenecks in a multi-user environment. For example, MP CONNECT could create multiple sessions that each use the SPDE engine to read input from a common data set with much less I/O contention.

Other benefits of the SPDE engine are that it will do an implicit sort of the data that is returned to the application if a BY statement is present. The disk copy of data does not get sorted, only the data that is returned to the application in memory. In addition, WHERE processing will make use of multiple indexes when possible; the Base engine uses only one.

The SPDE engine evolved from the SPD Server product; therefore, its feature set is derived from SPD Server. This also means that data sets produced by the SPDE engine and SPD Server are interchangeable. SPD Server supports a client/server environment requiring multiple SAS sessions. It also provides more functionality than the SPDE engine. However, the need to bring support of partitioned data into BASE SAS resulted in the creation of the SPDE engine. Unlike SPD Server, the engine runs entirely in the same SAS process or session as the rest of your SAS job. It is not the default engine for Version 9 and must be specified with an engine name of SPDE on a libname statement. The SPDE engine also does not support all of the features of the Base engine. It is a goal to support Base engine functionality in the SPDE engine in a future release. However, the Base engine will continue to be supported because the data sets created by these two engines are not interchangeable.

	Base engine	SPDE engine
Block reads	yes	yes
Partitioned data	no	yes
Parallel WHERE	no	yes
Locking	record	member
Number of cols	<32K	>32K
Number of rows	2**31 -32 bit	2**63 - all
Use multiple indexes	no	yes
SAS catalogs	yes	no
Integrity const.	yes	no
Audit trail	yes	no

Table 1: Comparison of Base and SPDE engines

Table 1 compares several SAS data processing features with respect to their availability in the Version 9 Base engine and the SPDE engine.

SAS/ACCESS ENGINES

Several of the SAS/ACCESS engines have been modified in Version 9 to take advantage of a new I/O subsystem that allows reading entire blocks of data instead of reading data just one record at a time. This block read capability reduces I/O bottlenecks and allows procedures to read data as fast as they are able to process it. The following SAS/ACCESS engines support this functionality:

- Oracle
- Sybase,
- DB2 (Unix and PC)
- ODBC
- SQL Server
- Teradata

These DBMS engines now have the capability to access DBMS data in parallel by using multiple threads to the parallel DBMS server. With this technology these SAS/ACCESS engines can deliver data to SAS much faster than before. Coupling the threaded SAS procedures discussed above with these SAS/ACCESS engines provides even greater gains in performance.

One of the limitations to the amount of scalability that can be seen with the SAS/ACCESS engines is the efficiency of parallelization implemented in the DBMS itself. In the SAS environment there are options available on the libname statement that enable tuning of the threaded implementation within the SAS/ACCESS engines themselves.

SCALABLE SAS PROCEDURES

From both in-house as well as customer experience, several SAS procedures have been identified as performance critical, especially for long-running applications. In Version 9 some of these procedures have been modified to take advantage of faster I/O and SMP hardware. With the appropriate hardware configuration and these enhanced procedures, bottlenecks in your long-running applications can be minimized.

A new I/O subsystem has been introduced in Version 9 that allows reading entire blocks of data instead of

reading data just one record at a time. The block read capability reduces I/O bottlenecks and allows procedures to read data as fast as they are able to process it. This functionality is supported by the Base engine, several of the SAS/ACCESS engines, as well as by the SPDE engine. Combined with the engine's ability to read partitioned data, it is now possible for SAS procedures to read blocks of data in parallel. The procedures that have been modified so far in Version 9 to take advantage of this high speed I/O include:

- in BASE SAS
  - PROC SORT,
  - PROC SUMMARY
- in SAS/STAT
  - PROC REG
- in Enterprise Miner
  - PROC DMREG
  - PROC DMDB
  - PROC DMINE

More procedures are being converted to use this new I/O subsystem.

For SAS procedures (or portions of the procedures) that are CPU intensive, internal modifications have been made to take advantage of threads. Multiple threads are used to parallelize the computational portions of the procedures. Provided you run these procedures on SMP hardware, you will receive faster results without making any changes to your SAS programs.

SPD SERVER

The *Scalable Performance Data Server (SPD Server)* is a client/server, multi-user data server designed to optimize storage and to speed the processing of large SAS data sets. SPD Server does this by parallelizing many of the SAS I/O functions such as WHERE processing and INDEX creation over multiple data partitions in the same way that the SPDE engine works. However, SPD Server extends parallel capabilities to include GROUP BY processing and SQL passthru. SPD Server requires an SMP machine and is designed to use all resources available on the machine to achieve maximum scalability. The maximum benefit with SPD Server is gained when it is run on a machine with:

- multiple cpus
- multiple I/O channels
- multiple disks
- large amount of data to be partitioned

In addition to parallel processing capabilities, SPD Server also provides security features including userid/password validation and ACL file security. It also provides backup and recovery facilities.

In order to use SPD Server you must invoke two sessions on the server machine: one to run a *name server* and one to run the actual *data server*. When a client connects to SPD Server through the use of a libname statement, a proxy SAS session (*spdsbase)* is automatically started to process all client requests. SPD Server 3.0 supports server sessions on HP-UX, Solaris, AIX, Compaq's Digital Unix, and Windows NT. The client applications can execute in sessions on the same or different machines. The client applications connect to the server using syntax on the libname statement. Therefore, once the server environment has been initiated, code modifications are limited to the libname statement(s) in the client application.

SPD Server is not new to Version 9. It was initially released in conjunction with Release 6.12 of the SAS System. SPD Server provides a high performance data store of very large SAS data sets. Therefore, it is particularly suited as part of a data warehousing solution where the SAS system is being used to construct, manage and analyze enterprise-wide datamarts.

OPEN METADATA SERVER

The Open Metadata Server provides a set of common repositories for storing metadata required by various SAS solutions. The Open Metadata Server retrieves metadata from one or more repositories as multiple SAS solutions request it. Therefore, performance is a critical issue. To this end, the Open Metadata Server makes use of threads to enable the best response time for serving metadata to any number of clients making requests. The resource allocation of the Open Metadata Server is controlled through a configuration file.

OLAP SERVER

The Online Analytical Processing Server (OLAP Server) is a multi-user, scalable, online analytical processing server that can be used to store and access large volumes of data while maintaining system performance. The OLAP Server allows data to be stored as a cube and then analyzed from many different view points or dimensions. The data can be

multidimensional (MOLAP) or a hybrid combination (HOLAP). Because the volume of data stored in these databases and the number of users making requests to the server at any one time can be very large, it is important that the software used to deliver the data be scaled for maximum performance.

The Version 9 OLAP Server makes use of many new features to provide maximum performance. It makes use of the same storage technology used by the SPDE engine for storing and accessing the cubes. The OLAP Server is also threaded to provide parallel I/O to partitioned data and parallel processing of the data as it is read. The amount of threading that the server actually performs can be user-controlled by an option in the server configuration as well as in the administrator user-interface. These enhancements result in a scalable OLAP Server that provides faster access and processing of your data.

	Level of parallelization	User control or tuning	SAS Version initially available	Type of scalability
MP CONNECT	separate SAS sessions	signon, rsubmit, etc. syntax	Version 8.0	Scale up (SMP machine) Scale out (across network)
SPDE ENGINE	threading within the engine	libname options	Version 9	Scale up (SMP machine)
SAS/ACCESS ENGINES	threading within the engine and DBMS	libname options	Version 9	Scale up (SMP machine)
SCALED SAS PROCEDURES	threading within a SAS procedure	global options	Version 9	Scale up (SMP machine)
SPD SERVER	threading within a SAS process	invoke server environment, libname options	Version 6.12	Scale up (SMP machine)
OPEN METADATA SERVER	threading within a SAS process	config file	Version 9	Scale up (SMP machine)
OLAP SERVER	threading within a SAS process	global options or administrator GUI	Version 9	Scale up (SMP machine)

Table 2 - Scalability features across solutions.

Table 2 summarizes the areas that address performance through scalability in Version 9. The *level of parallelization* refers to whether a separate SAS session is invoked to execute in parallel or whether multiple threads are used within the same SAS session to achieve parallelization. *User control or tuning* refers to what an applications programmer would need to add or modify in a SAS program in order to take advantage of a feature or tune a feature to maximize the benefit in performance. *SAS Version initially available* indicates when the scalable features of each of the solutions were initially available. *Type of scalabiltiy* refers to whether a feature is able to scale up - exploit scalable hardware

on a single machine, or also scale out - exploit hardware across a network.

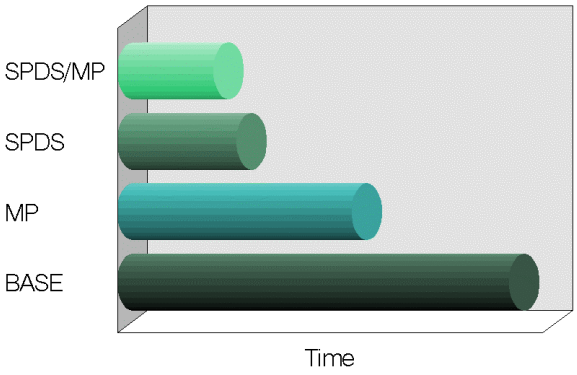
SCALE UP AND SCALE OUT

As stated earlier, whether to scale up or scale out is not a mutually exclusive choice. In fact, many applications lend themselves to a combination of the two strategies. To illustrate this, a couple of scenarios will be described that combine two or more of the SAS scalability solutions in a systematic approach to illustrate the increased performance gains that are possible.

The first scenario starts with an application implemented with Base SAS. The application will then be modified to incorporate MP CONNECT and show the gains in performance. Then it will be modified to use an SPD Server solution. Lastly the application will be modified to combine MP CONNECT with SPD Server for a maximum gain in performance.

The business case involves the following steps:

- sort data a and data b
- merge data a and data b, create data c
- sort data d
- input data d, create data e and data f
- merge data c and data e



Graph 1: Base, MP CONNECT, SPDS solutions

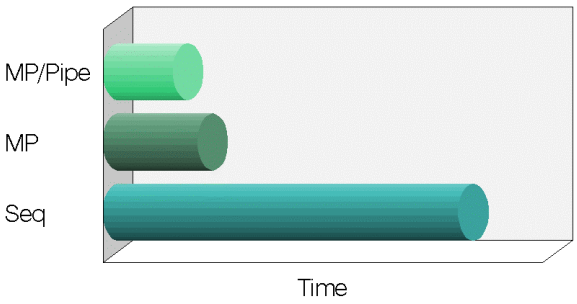
Graph 1 shows the gain in performance with each iteration of the application. The *BASE* result came from running the sorts and data steps serially in a single SAS session. The *MP* result came from adding statements to the job to create two SAS sessions to execute in parallel. The first SAS session performed the sort of data a and b and the merge to create c. The second session performed the sort of d and the creation of data e and f. Upon completion of these two parallel tasks, the merge of data c and e took place. The *SPDS* result came from modifying the application to use SPD Server. First, the SPD Server environment was initialized and then a client SAS session was invoked where the application was modified to use a libname that pointed to the SPD Server. Big gains were seen here because the explicit PROC SORT steps could be eliminated in favor of the implicit sort available with SPD Server. Because the data sets were sorted on a single BY variable, SPD Server was able to perform an implicit sort of the data and push it to the server to be executed. The SPDE engine provides the same implicit sort capability therefore, similar improvement would be seen by using the SPDE engine in place of SPD Server. The biggest jump in performance came by combining the multiple process capabilities of MP CONNECT with the implicit threaded sort capabilities of SPD Server. The *SPDS/MP* result was obtained by using MP CONNECT to create two sessions. The first session used SPD Server to implicitly sort data a and b while merging them into data c. The second session used SPD Server to implicitly sort data d while creating data e and f. Upon completion of these two parallel tasks, the merge of data c and e took place.

The final implementation of this scenario using MP CONNECT in conjunction with SPD Server ran in a little less than one fourth the time of the initial implementation. These results were obtained with a specific application and hardware configuration and will vary depending on your application characteristics and hardware.

The second scenario also starts with an application implemented using just Base SAS. The application will then be modified to incorporate MP CONNECT and show the gains in performance. Then it will be modified to combine MP CONNECT with the new piping functionality for a maximum gain in performance.

The business case involves the following steps:

- read 5 large raw data files
- data step processing of each input file to create 5 SAS data sets
- sort each resulting data set



Graph 2: Base, MP CONNECT, Piping solutions

Graph 2 shows the gain in performance with each iteration of the application. The *Seq* result represents using Base SAS to read in the first raw data file, perform the data step processing, sort the file and then repeat this series of steps for the next four input files. The *MP* result depicts incorporating MP CONNECT to launch five separate SAS sessions to run in parallel with each session reading an input file, creating a data set, and then sorting the data set. The *MP/Pipe* result illustrates the use of the piping capability in conjunction with MP CONNECT to overlap the I/O of the data step and the PROC SORT. In this final implementation, ten parallel sessions are spawned; five sessions to each read a data file and create a SAS data set, with five additional sessions to each run PROC SORT. Rather than writing the data set to disk, each data step running in one session pipes its output to the appropriate session running PROC SORT so that the execution of the data step and PROC SORT can overlap. In addition, the piping feature eliminates the need to write the intermediate data set to disk reducing disk space requirements.

The final implementation of this scenario using MP CONNECT in conjunction with Piping ran in about one fifth the time of the initial implementation. These results were obtained with a specific application and hardware configuration and will vary depending on your application characteristics and hardware.

SUMMARY

This paper introduces several ways that the SAS System is addressing performance through scalability to exploit the full processing power of your SMP machines as well as the processing resources available across your networks. By combining scalable SAS software with the right hardware configuration, you can find the right balance to achieve true scalability of your critical applications. You can do this by taking advantage of:

- partitioned data
- parallel I/O
- parallel multi-threaded computation
- data piping
- parallel sessions

The features and products that are available with SAS Version 9 are only the beginning of an evolutionary path to deliver even more performance enhancements with future releases.

REFERENCES

Olson, D. and Ray, R. (2001), "Version 9: Scaling the Future," *Proceedings of the Twenty-sixth Annual SAS Users Group International Conference.*

"Scalable Performance Data Server, Version 2.0," *Twenty-third Annual SAS Users Group International Conference.*

ACKNOWLEDGEMENTS

Many people have contributed to the products and features described in this paper. They include but are not limited to:

MP CONNECT
Cheryl.Doninger@sas.com
Renee.Palmer@sas.com

SPDE Engine
Billy.Clifford@sas.com

SAS/ACCESS Engines
Howard.Plemmons@sas.com
Nancy.Wills@sas.com

Scalable SAS Procedures
Robert.Ray@sas.com
Robert Cohen@sas.com

SPD Server
Ami.Ghosh@sas.com
Dan.Sargent@sas.com

Open Metadata Server
Craig.Rubendall@sas.com

OLAP Server
Duane.Ressler@sas.com

AUTHOR CONTACT INFORMATION

Cheryl Doninger
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
(919) 531-7941
Cheryl.Doninger@sas.com