Paper 401-2013

High-Performance Statistical Modeling

Robert A. Cohen and Robert N. Rodriguez, SAS Institute Inc.

ABSTRACT

The explosive growth of data, coupled with the emergence of powerful distributed computing platforms, is driving the need for high-performance statistical modeling software. SAS has developed a group of high-performance analytics procedures that perform statistical modeling and model selection by exploiting all the cores available—whether in a single machine or in a distributed computing environment. This paper describes the various execution modes and data access methods for high-performance analytics procedures. It also discusses the design principles for high-performance statistical modeling procedures and offers guidance about how and when these procedures provide performance benefits.

INTRODUCTION

This paper introduces high-performance statistical modeling procedures that are included in SAS/STAT® 12.3. The paper uses the term "HPA procedures" to refer generically to high-performance analytics procedures.

The paper is organized into two major parts. Part I describes general considerations that apply to all HPA procedures, and it contains into the following sections:

- "EXECUTION MODES" describes how HPA procedures can run on a single server or in a distributed computing environment. This section also introduces terminology that is used throughout the paper.
- "AVAILABILITY" discusses how the HPA procedures are licensed.
- "DATA ACCESS FEATURES" describes the specialized "big data" access functionality that is integrated into all HPA procedures. This functionality enables HPA procedures to run in an "alongside-the-database" mode, so that the analysis is performed on the same hardware where the database resides.
- "SYMMETRIC AND ASYMMETRIC MODES" describes how HPA procedures can execute in a mode where computations are performed in a distributed computing environment that is separate from the database appliance that houses the data. This is accomplished by efficiently moving data in parallel between multiple machines.
- "IN-MEMORY COMPUTATION" describes how the vast amount of memory available in distributed computing environments enables HPA procedures to execute entirely in memory.

Part II narrows the focus to high-performance statistical modeling procedures and is organized into the following sections:

- "DESIGN PRINCIPLES" describes the design considerations that are followed in developing the high-performance statistical modeling procedures.
- "HIGH-PERFORMANCE STATISTICAL MODELING PROCEDURES" briefly describes the high-performance statistical modeling procedures that are included with SAS/STAT.
- "PERFORMANCE AND SCALABILITY" presents examples that illustrate how you can use various execution modes of high-performance statistical modeling procedures to scale to a wide range of problem sizes.

PART I: GENERAL CONSIDERATIONS

This part of the paper explains terminology and execution modes that apply to all high-performance analytics procedures. You can find specific information about high-performance statistical modeling procedures in Part II.

EXECUTION MODES

The fundamental distinction between HPA procedures and traditional (MVA) procedures is the way in which they execute. HPA procedures are engineered to run either on a single server or in a distributed mode that uses a cluster of computers. In contrast, MVA procedures can execute only on a single server.

A second distinction is that all HPA procedures are multithreaded and can exploit all the cores available, whether in a single machine or in a distributed computing environment. In contrast, not all MVA procedures are multithreaded.

Single-Machine Mode

An HPA procedure is said to execute in single-machine node when all the computation is done on the server where SAS is installed. Single-machine mode is also called "client mode," but this paper uses the term single-machine mode to emphasize that in this mode all computations are performed on a single machine.

An important feature of all HPA procedures when they run in single-machine mode is that they are engineered to solve analytic tasks in parallel by using concurrently scheduled threads that exploit the multiple processors that are ubiquitous in current-generation computers. The procedure determines the number of concurrent threads based on the number of CPUs (cores) on the machine. Depending on the analytic task, HPA procedures use different methods to map the number of cores to the number of concurrent threads. Using one thread per core is not uncommon for procedures that implement data-parallel algorithms.

You might sometimes see single-machine mode referred to as SMP (symmetric multiprocessing) mode. Symmetric multiprocessing is a common server architecture in which all processors have uniform access to the main memory on the server. However, because some modern servers use a NUMA (nonuniform memory access) architecture, the term single-machine mode is technically more accurate.

Distributed Mode

When HPA procedures execute in distributed mode, the SAS server functions as a client that communicates with and drives specialized SAS software that runs on a tightly integrated cluster of homogeneous machines. The data are distributed across the machines in this cluster, and the massive computing power of the cluster is brought to bear to solve a single large analytic task.

When an HPA procedure executes in distributed mode, multiple concurrently scheduled threads are used on each machine in the cluster, and the number of threads used on each machine corresponds to the number of cores on the machine. For example, in a cluster of 16 machines, each with 24 cores, a total of 384 (16*24) threads execute simultaneously to solve the analytic task.

You might sometimes see distributed mode referred to as MPP (massively parallel processing) mode, because this term emphasizes that computations are done simultaneously on multiple computers, each of which uses multiple threads. The umbrella term "grid computing" is also sometimes used to describe distributed mode. However, grid computing can also refer to the process by which a cluster of loosely connected computers, each running an independent SAS session, executes multiple independent tasks simultaneously. You can use the MP CONNECT functionality in SAS/CONNECT® software to implement this type of grid computing, and you can use SAS® Grid Manager software to manage such a grid.

You might also see the terms "appliance" and "blade server" used to describe a tightly integrated homogeneous cluster of computers that are arranged in units called racks. The individual computers in each rack are called "nodes" or "blades." Database appliances are appliances that include database software to manage data that are stored in distributed fashion across the nodes of the appliance. It is not necessary that database software run on an appliance in order for HPA procedures to be able to run on the appliance.

AVAILABILITY

Prior to SAS 9.4, you could obtain HPA procedures only by licensing a product called the SAS® High-Performance Analytics Server. This product included procedures for data mining, statistical modeling, time series analysis, and text mining. You could not license only a subset of procedures for the particular functionality that you might need. Furthermore, you could not access these procedures in single-machine mode without licensing them to run also in distributed mode.

To enable you to obtain just the functionality that you need, SAS-High Performance Analytics Server has now been split into products that can be individually licensed to run in distributed mode. Each of these products is associated with an existing MVA product as shown in the Table 1.

Table 1: SAS High-Performance Analytics Products and MVA Products

SAS High-Performance Analytics Product	MVA Product
SAS® High-Performance Statistics	SAS/STAT®
SAS® High-Performance Econometrics	SAS/ETS®
SAS® High-Performance Optimization	SAS/OR®
SAS® High-Performance Forecasting	SAS® High-Performance Forecasting
SAS® High-Performance Data Mining	SAS® Enterprise Miner™
SAS® High-Performance Text Mining	SAS® Text Miner

Furthermore, the procedures in the SAS High-Performance Analytics products are included with the associated MVA product, and you can run these procedures in single-machine mode without licensing the high-performance product. However, you do need to license the SAS High-Performance Analytics product to run its procedures in distributed mode.

For example, if you license SAS/STAT, you can run in single-machine mode the HPGENSELECT, HPLMIXED, HPLOGISTIC, HPNLMOD, HPREG, and HPSPLIT procedures. For more information about these procedures, see the section "HIGH-PERFORMANCE STATISTICAL MODELING PROCEDURES" on page 6. To run these procedures in distributed mode, you must license SAS High-Performance Statistics software in addition to SAS/STAT software.

DATA ACCESS FEATURES

The following data access methods are supported by HPA procedures when they run in distributed mode:

- Client-data (or local-data) method. The data are stored on the machine where SAS is installed (referred to as the "client"). When the procedure runs, the data are moved to the distributed computing environment by the high-performance analytics infrastructure.

  Note: The client-data method should not be confused with single-machine mode, in which the data remain on the client machine and the computations are performed on the client machine.

- Alongside-the-database method. The data are stored in the distributed database management system (DBMS) and are read in parallel from the DBMS into a SAS analytic process that runs on the database appliance.

- Alongside-HDFS method. The data are stored in the Hadoop Distributed File System (HDFS) and are read in parallel from the HDFS.

- Alongside-LASR mode. The data are loaded from a SAS® LASR™ Analytic Server that runs on the appliance.

The following subsections provide more detail.

Alongside-the-Database Execution

High-performance analytics procedures interface in a unique way with the DBMS on the appliance. If the input data are stored in the DBMS and the appliance on which the analyses are performed is the appliance that houses the data, HPA procedures create a distributed computing environment in which an analytic process is co-located with the nodes of the DBMS where the data are stored. Data then pass from the DBMS to the analytic process on each node. This avoids having to move data across the network and possibly back to the client machine. Instead, the data pass locally between the processes on each node of the appliance.

Because the analytic processes on the appliance are separate from the database processes, this mode of execution is referred to as alongside-the-database execution. It is different from in-database execution, used by SAS® In-Database Analytics, where the analytic code executes in the database process.

In general, when you have a large amount of data, you can obtain the best performance from HPA procedures if execution is alongside the database.

Alongside-HDFS Execution

Running HPA procedures alongside HDFS shares many features with running alongside the database. The procedures use the distributed computing environment, in which an analytic process is co-located with the nodes of the cluster. Data then pass from HDFS to the analytic process on each node of the cluster.

Alongside-LASR Execution

When HPA procedures execute in distributed mode alongside a SAS LASR Analytic Server, the data are preloaded in distributed form in memory that is managed by the SAS LASR Analytic Server. The data on the nodes of the appliance are accessed in parallel in the process that runs the server, and they are transferred to the process where the HPA procedure runs.

In general, each HPA procedure copies the data to memory that persists only while that procedure executes. Hence, when an HPA procedure runs alongside a LASR Analytic Server, both the procedure and the server have a copy of the subset of the data that are used by the procedure.

The advantage of running HPA procedures alongside a LASR Analytic Server (as opposed to running alongside a DBMS table or alongside HDFS) is that the initial transfer of data from the server to the HPA procedure is a memory-to-memory operation that is faster than the disk-to-memory operation that occurs when the procedure runs alongside a DBMS or HDFS. Alongside-LASR execution can result in improved performance when the cost of preloading a table into the server is amortized over multiple uses of these data in separate runs of HPA procedures.

SYMMETRIC AND ASYMMETRIC MODES

When you run HPA procedures alongside the database or alongside HDFS, you can run them in either symmetric mode or in asymmetric mode, as described in the following sections.

Symmetric Mode

HPA procedures are said to be operating in symmetric mode when they run alongside the database and alongside HDFS on the appliance that houses the data and when the number of nodes used for the computations is the same as the number of nodes that hold the data.

When HPA procedures run in symmetric mode, the data appliance and the computing appliance must be the same appliance. The HPA procedures—together with a SAS® Embedded Process, which reads and writes data from the DBMS—execute in a SAS process that runs on the same hardware where the DBMS process executes. This is called symmetric mode because the number of nodes on which the DBMS executes is the same as the number of nodes on which the HPA procedures execute. The initial data movement from the DBMS to the HPA procedure does not cross node boundaries.

Asymmetric Mode

HPA procedures can also run alongside-the-database and alongside-HDFS in an asymmetric mode. The primary reason for providing the asymmetric mode is to enable you to manage and house your data on one appliance (the data appliance) and to run HPA on a second appliance (the computing appliance). In asymmetric mode, the data appliance and the computing appliance are usually distinct appliances. This mode is called asymmetric because the number of nodes on the data appliance does not need to be the same as the number of nodes on the computing appliance. The HPA procedures execute in a SAS process that runs on the computing appliance. The DBMS and a SAS Embedded Process run on the data appliance. Data are requested by a SAS data feeder that runs on the computing appliance and communicates with the SAS Embedded Process on the data appliance. The SAS Embedded process transfers the data in parallel to the SAS data feeder that runs on each of the nodes of the computing appliance. You can also run in asymmetric mode on a single appliance that functions as both the data appliance and the computing appliance. This enables you to run alongside-the-database or alongside-HDFS, where computations are performed on a set of nodes that is different from the set of nodes on which the data reside.

IN-MEMORY COMPUTATION

The combined amount of memory available on distributed computing appliances enables you to hold large volumes of data in memory. Appliances are commonly configured with nodes that have 256 GB of memory. This enables HPA procedures to hold many terabytes of data in memory on even moderately sized appliances. After the data have been loaded into memory, the entire computation takes place in memory, which yields substantial performance benefits when algorithms require multiple passes through the data.

In single-machine mode, the amount of memory might not be large enough to hold the entire data set to be analyzed in memory. In such cases, utility files, which are stored on disks, are used where needed so that the computation can be completed. However, this use of disks during a computation can significantly degrade the performance of an algorithm. When your data routinely exceed the amount of memory available on your single server, moving your computation to an appliance with its much larger memory space will yield greatly improved performance.

PART II: HIGH-PERFORMANCE STATISTICAL MODELING

The second part of this paper provides information about the design and performance features of high-performance statistical modeling procedures.

DESIGN PRINCIPLES

This section explains eight design principles for the development of high-performance statistical modeling procedures.

All HPA procedures are designed according to the following principles:

1. HPA procedures need to be able to access and analyze data in distributed computing environments.
2. HPA procedures need to be able to perform analyses in single-machine mode by using the same syntax that is used for distributed mode.

   An identical syntax for single-machine and distributed modes (other than options that specify the mode) gives you an easy upgrade path as the size of your data grows. You can develop applications on a single machine and then use the same procedure statements without modification in distributed mode.

In addition, high-performance statistical modeling procedures are designed according to the following principles:

3. The principal goal is building accurate predictive models.

   The term “big data” has become popular for describing large volumes of data. Many big data problems are characterized by millions or even billions of observations, and in many situations these data contain thousands of variables that can be used to build predictive models. Accomplishing this task by

exploiting massive data and powerful distributed computing infrastructures is the main motivation for developing these procedures.

4. The procedures should include methods for performing model selection, dimension reduction, and identification of important variables whenever this is appropriate for the analysis.

   For modeling scenarios in which the number of variables can be large, it is important to be able to perform variable selection or dimension reduction to obtain parsimonious models that generalize well for prediction of future data.

5. The procedures for predictive modeling should not need to implement computationally expensive methods for computing inferential statistics.

   Statistical methods for making inferences about populations based on small data samples are less important in big data situations, where the sample is very large or even consists of the entire population. Expending substantial amounts of time to compute inferential statistics that are not useful for building large-scale predictive models detracts from the ability of these procedures to analyze massive data sets.

6. The syntax for functionality that is common to high-performance statistical modeling procedures needs to be consistent; this takes precedence over consistency with the syntax of traditional SAS/STAT procedures.

   All HPA procedures have common options that control the modes in which the procedures operate. Moreover, several high-performance statistical modeling procedures share common functionality such as model selection. A consequence this syntax consistency among the high-performance statistical modeling procedures is that the syntax of HPA procedures sometimes varies from the syntax of traditional SAS/STAT procedures. For example, the high-performance statistical modeling procedures that build regression models share a common SELECTION statement for controlling model selection, as opposed to MODEL statement options that are used in the REG, LOGISTIC, and GLMSELECT procedures.

7. There need not be a one-to-one correspondence between the syntax and functionality of high-performance statistical modeling procedures and that of traditional SAS/STAT procedures.

   Developing new high-performance modeling procedures provides an opportunity to consolidate similar functionality that is drawn from several SAS/STAT procedures into a single high-performance statistical modeling procedure. An example is PROC HPREG, which has features drawn from the GLMSELECT, GLM, and REG procedures.

   Moreover, high-performance statistical modeling procedures can implement functionality that is not found in traditional SAS/STAT procedures. For example, both PROC HPGENSELECT and PROC GENMOD fit generalized linear models, but PROC HPGENSELECT focuses on model selection, which is not available in PROC GENMOD. In other cases, high-performance statistical modeling procedures can implement just the subset of a traditional SAS/STAT procedure's functionality that is needed for predictive modeling. For example, PROC HPLOGISTIC does not implement many of the small-sample and inferential features that are available in PROC LOGISTIC.

8. New development in SAS/STAT software will be done using the high-performance framework whenever this is a viable approach for statistical modeling.

   As data volumes grow, there will be an increasing need for statistical modeling software to analyze these data. New procedures in SAS/STAT are being developed as high-performance procedures when this approach is relevant and feasible for analyzing big data. Two such examples in SAS/STAT 12.3 are the HPGENSELECT procedure and the HPSPLIT procedure.

HIGH-PERFORMANCE STATISTICAL MODELING PROCEDURES

The following subsections describes the high-performance statistical modeling procedures that are included with SAS/STAT 12.3.

The HPGENSELECT Procedure

The HPGENSELECT procedure performs model selection for generalized linear models and fits generalized linear models for a broad class of distributions. The procedure permits a variety of link functions; it can fit zero-inflated models for count data and multinomial models for ordinal and nominal data that have more than two response categories.

The HPLMIXED Procedure

The HPLMIXED procedure fits mixed linear models to data that have complex covariance structures. The procedure implements techniques that are commonly used to model clustered or repeated data, where observations within a cluster or subject are correlated. The HPLMIXED procedures provides a variety of covariance structures: variance components, compound symmetry, unstructured, AR(1), Toeplitz, and factor analytic. Models are fit by either maximum likelihood or restricted maximum likelihood (REML), using special dense and sparse algorithms that take advantage of distributed and multicore computing environments.

The HPLOGISTIC procedure

The HPLOGISTIC procedure performs model selection for a broad range of logistic regression models. The procedure permits several link functions and can handle ordinal and nominal data that have more than two response categories (multinomial models).

The HPNLMOD Procedure

The HPNLMOD procedure uses either nonlinear least squares or maximum likelihood to fit nonlinear regression models. The procedure enables you to specify the model by using SAS programming statements. This gives you greater flexibility in modeling the relationship between the response variable and independent (regressor) variables than procedures that use a more structured MODEL statement.

The HPREG Procedure

The HPREG procedure performs model selection for ordinary linear least squares regression models. The procedure supports general linear models for independently and identically distributed data. The models can contain main effects that consist of both continuous and classification variables, and they can contain interaction effects of these variables. The procedure offers extensive capabilities for customizing the model selection by using a wide variety of selection and stopping criteria, from traditional and computationally efficient significance-level-based criteria to more computationally intensive validation-based criteria. The HPREG procedure also provides a variety of regression diagnostics that are conditional on the selected model.

The HPSPLIT Procedure

The HPSPLIT procedure creates a decision tree model. The procedure can create decision tree models by using a number of methods to grow and prune the tree. The output that is created by the HPSPLIT procedure can be used to tune models and determine input variable importance.

Utility High-Performance Procedures

The following high-performance utility procedures are included with all MVA products that include HPA procedures.

HPBIN	performs bucket, winsorized, and pseudo–quantile binning.
HPCORR	computes Pearson correlations.
HPDMDB	computes metadata and summaries of input data.
HPDS2	executes DS2 statements in parallel.
HPIMPUTE	imputes missing values for numeric data.
HPSAMPLE	performs simple and stratified sampling.
HPSUMMARY	computes summary statistics.

PERFORMANCE AND SCALABILITY

This section provides information about the performance and scalability of high-performance statistical modeling procedures. Examples that feature model selection with simulated data are used to illustrate the performance and scalability of these procedures.

Simulated Data

The examples in this section all use a simulated data set named sgf2013 that is generated from SAS code that is provided in "APPENDIX" on page 18. These data contain three response variables, **yNormal**, **yBinary**, and **yPoisson**, which are modeled by using linear regression in PROC HPREG, logistic regression in PROC HPLOGISTIC, and Poisson regression in PROC HPGENSELECT. In order to demonstrate the model selection capabilities of these procedures, the responses have all been constructed to depend systematically on the same subset of the complete set of regressors, which are described in Table 2.

Table 2: Complete Set of Regressors

Regressor Name	Type	Number of Levels	In True Model
xIn1–xIn20	Continuous		Yes
xSubtle	Continuous		Yes
xTiny	Continuous		Yes
xOut1–xOut80	Continuous		No
cIn1–cIn5	Classification	From two to five	Yes
cOut1–cOut5	Classification	From two to five	No

The labels **In** and **Out** that are part of the variable names make it easy to identify when selected models succeed or fail in capturing the true underlying models. The regressors that are labeled **xSubtle** and **xTiny** are regressors that are predictive, but whose influence is substantially smaller than that of the other regressors in the true model. Identifying these two regressors requires a large number of simulated observations.

The following examples use subsets of sgf2013 whose number of observations ranges from 50,000 to 50 million. When the data set contains 50 million observations, it is 45 GB in size.

Performance Comparisons with Traditional SAS/STAT Procedures

In many situations, you can carry out the same analysis with a high-performance statistical modeling procedure and a traditional SAS/STAT procedure on a single machine that has multiple cores. When the comparison is made with a traditional SAS/STAT procedure that does not support multithreading, then for all but very short running analyses you will realize a performance gain by using the high-performance procedure. However, when the traditional SAS/STAT procedure supports multithreading, you will typically get similar—though not identical—performance.

The following statements fit the same logistic regression model and were executed on a desktop PC that has four CPUs. Note that **x:** and **c:** are shorthand for regressors whose names begin with **x** and **c**, respectively.

```
proc logistic data=sgf2013(obs=1000000);
  class c: / param=glm;
   model yBinary = x: c:;
run;

proc hplogistic data=localLib.sgf2013(obs=1000000);
   class c: ;
   model yBinary = x: c:;
run;
```

The HPLOGISTIC procedure, which exploits all of the CPUs, ran in 31 seconds, about 3.7 times faster than the LOGISTIC procedure, which ran in 116 seconds.

The following statements lead to a similar comparison by using PROC GENMOD and PROC HPGENSELECT to fit the same Poisson regression.

```
proc genmod data=sgf2013(obs=1000000);
   class c:;
   model yPoisson = x: c: /dist=Poisson;
run;

proc hpgenselect data=sgf2013(obs=1000000);
   class c:;
   model yPoisson = x: c: /dist=Poisson;
run;
```

In this test, PROC HPGENSELECT ran in 35 seconds, about 11.6 times faster than PROC GENMOD, which ran in 407 seconds. Even when limited to using just a single thread, PROC HPGENSELECT ran in 112 seconds, which is still significantly faster than PROC GENMOD.

Scalability in Single-Machine Mode

Because all high-performance statistical modeling procedures are multithreaded, you will realize performance gains when you run these procedures in single-machine mode on a server that has multiple CPUs. In most cases, you will see scalability for a given test as the number of threads begins to increase from one, but after a point the performance gains will start to diminish.

There are two reasons for this drop-off in scalability: The first reason—known as "Amdahl's Law"—is that a small component of every job executes serially. Even if the time taken by the rest of the computation scales perfectly with increasing utilization of multiple CPUs, this serial component makes an ever increasing contribution to the total run time. You can find a detailed discussion of this phenomenon in Cohen (2002).

A second reason for the drop-off in scalability is the memory bandwidth limitation of the hardware. As you increase CPU utilization, you place greater demands on the memory system that needs to deliver memory simultaneously to all the CPUs. To improve the speed at which memory can be accessed by the CPUs, modern computer architectures use a hierarchy of memory caches. Accessing memory that resides in a local cache is much faster than accessing main memory. However, when a CPU needs data that are not already in a cache, then a cache miss occurs and the requested memory needs to be retrieved from a higher level of cache or even from main memory. If a large number of cache misses occur and the CPUs have to wait for needed memory, you will experience a drop in scalability.

In general, the scalability that you observe depends on the specific test and the specific hardware on which the test is run. For example, tests that use PROC HPREG to perform linear regression usually exhibit less scalability than similarly sized tests that use PROC HPLOGISTIC to perform logistic regression. The reason is that the ratio of I/O time to computing time is much larger for linear regression than for logistic regression, and I/O is usually performed serially on a single machine.

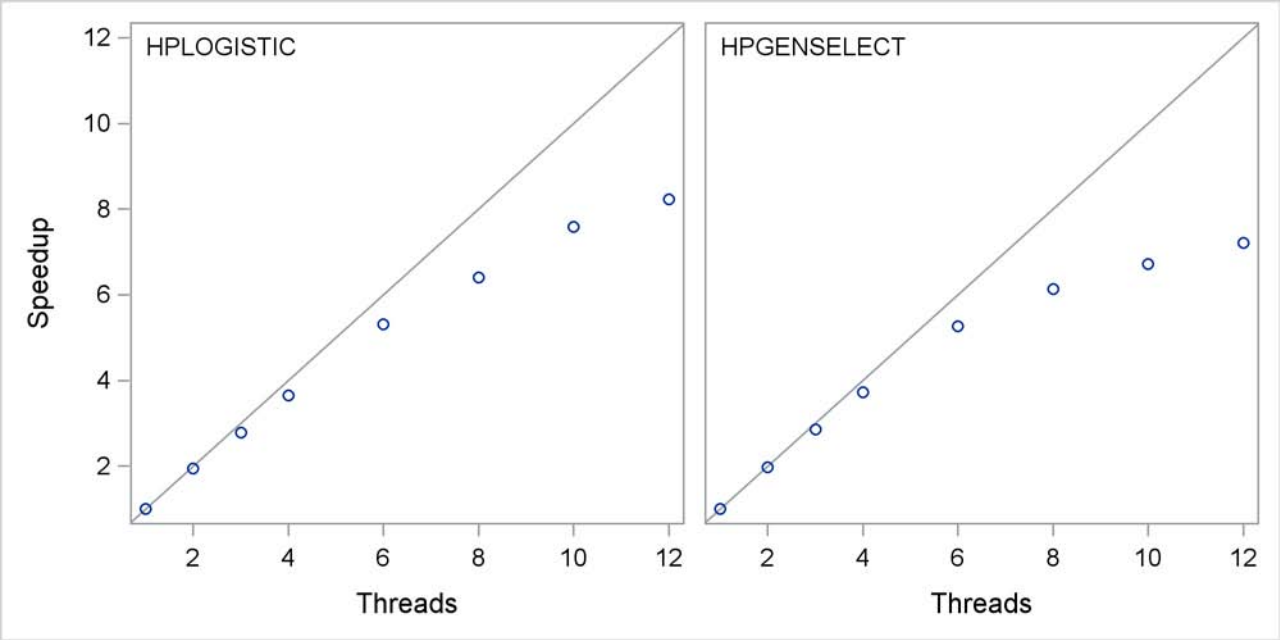
The following PROC HPGENSELECT and PROC HPLOGISTIC steps were run on a 12-CPU Windows server to generate the scalability plots that are shown in Figure 1:

```
proc hplogistic data=sgf2013(obs=1000000);
    class c: ;
    model yBinary = x: c: / dist=poisson;
    performance nthreads = &nThreads;
run;

proc hpgenselect data=sgf2013(obs=1000000);
    class c: ;
    model yPoisson = x: c: / dist=poisson;
    performance nthreads = &nThreads;
 run;
```

This test uses 1 million observations from the data set sgf2013. The macro variable *nThreads* is set to 1, 2, 3, 4, 6, 8, 10, and 12, and the elapsed times (denoted by $t_1, t_2, t_3, \ldots, t_{12}$) are recorded. The speedups s_i that are the y coordinates of the points in Figure 1 are defined by $s_i = t_1/t_i$.

Figure 1 Scalability on a 12-CPU Windows Server



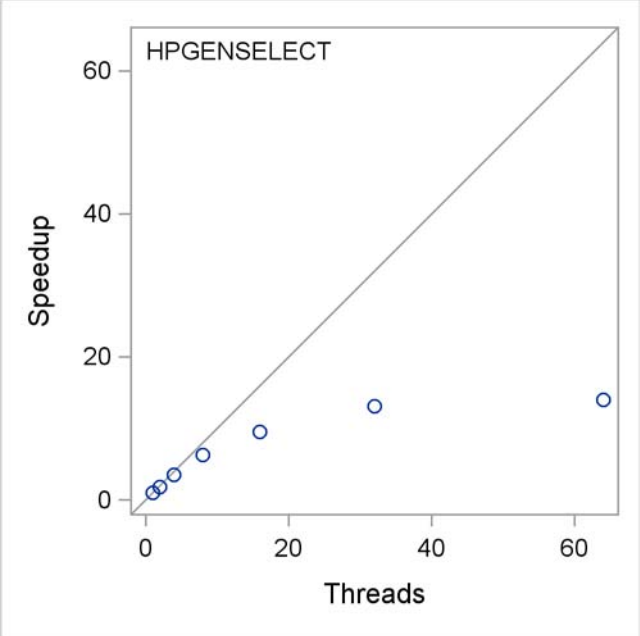
In Figure 1 the solid diagonal line represents perfect scalability, which is also called linear scalability. You see that these tests exhibit nearly linear scalability up to six threads, but then scalability starts tailing off.

Figure 2 shows a similar scalability test for the following PROC HPGENSELECT step, which was run on a 64-CPU UNIX server:

```
proc hpgenselect data=sgf2013(obs=1000000);
    class c: ;
    model yPoisson = x: c: / dist=poisson;
    selection method=stepwise(choose=sbc);
    performance nthreads = &nThreads;
 run;
```

These statements repeat the test with 1, 2, 4, 8, 16, 32, and 64 threads.

Figure 2 Scalability on a 64-CPU UNIX Server



Because this test includes stepwise model selection, the time spent loading data into memory is amortized by many passes though this memory as candidates for entry or removal from the model are evaluated at each step of the selection process. This amortization results in better scalability than would be observed if no model selection were done. This test exhibits reasonable scalability to about 16 threads, but then scalability drops off markedly as the number of threads increases.

Moving to Distributed Mode — Taking Advantage of Big Data

This subsection highlights the use of PROC HPGENSELECT to select a Poisson regression model. As the number of simulated observations increases, you can fit increasingly accurate models.

The following statements provide a starting point for the analysis in single-machine mode. The same statements are used by subsequent examples in this subsection, except that the number of observations that are used increases.

```
proc hpgenselect data=sgf2013(obs=50000);
    class c:;
    model yPoisson = x: c: / dist=poisson;
    selection method=stepwise(choose=sbc);
run;
```

Figure 3 shows the settings that are used in this analysis. The "Performance Information" table confirms that PROC HPGENSELECT executed on the client with four concurrent threads.

Figure 3 Single-Machine Mode

The HPGENSELECT Procedure

Performance Information

Execution Mode	On Client
Number of Threads	4

Model Information

Data Source	LOCALLIB.SGF2013_SIM
Response Variable	yPoisson
Class Parameterization	GLM
Distribution	Poisson
Link Function	Log
Optimization Technique	Newton-Raphson with Ridging

Selection Information

Selection Method	Stepwise
Select Criterion	Significance Level
Stop Criterion	Significance Level
Choose Criterion	SBC
Effect Hierarchy Enforced	None
Entry Significance Level (SLE)	0.05
Stay Significance Level (SLS)	0.05
Stop Horizon	1

Number of Observations Read	50000
Number of Observations Used	50000

Figure 4 shows the "Selection Summary" table and the "Selected Effects" table. The selected model contains most—but not all—of the true effects in the model, and it contains no nuisance effects. The "Selection Summary" table shows that if the selected model had been based solely on the preset entry and

stay significance levels (SLE and SLS), then the selected model would have contained several nuisance regressors. By using the data-dependent Schwarz-Bayesian criterion (SBC) to select the final model, you obtain a more parsimonious model.

Figure 4 Single-Machine Mode

The HPGENSELECT Procedure

Selection Summary

Step	Effect Entered	Number Effects In	SBC	p Value
0	Intercept	1	215578.975	.
1	cIn5	2	204825.696	<.0001
2	xIn20	3	197379.021	<.0001
3	cIn4	4	190320.737	<.0001
4	xIn19	5	183900.037	<.0001
5	xIn18	6	177814.623	<.0001
6	xIn17	7	172789.983	<.0001
7	xIn15	8	168336.001	<.0001
8	xIn16	9	163980.714	<.0001
9	xIn14	10	160426.937	<.0001
10	xIn13	11	157363.403	<.0001
11	cIn3	12	154747.767	<.0001
12	xIn12	13	152175.970	<.0001
13	xIn11	14	149973.877	<.0001
14	xIn10	15	148013.167	<.0001
15	xIn9	16	146395.314	<.0001
16	xIn8	17	145297.496	<.0001
17	xIn7	18	144388.018	<.0001
18	xIn6	19	143724.718	<.0001
19	cIn2	20	143171.628	<.0001
20	xIn5	21	142747.127	<.0001
21	xIn4	22	142428.790	<.0001
22	xIn3	23	142277.727	<.0001
23	cIn1	24	142208.806	<.0001
24	xIn2	25	142163.774	<.0001
25	xIn1	26	142130.903*	<.0001
26	xOut25	27	142134.214	0.0061
27	xOut24	28	142140.775	0.0391

* Optimal Value of Criterion

Selected Effects: Intercept xIn1 xIn2 xIn3 xIn4 xIn5 xIn6 xIn7 xIn8 xIn9 xIn10 xIn11 xIn12 xIn13 xIn14 xIn15 xIn16 xIn17 xIn18 xIn19 xIn20 cIn1 cIn2 cIn3 cIn4 cIn5

Will you get a better model if you use more data? The following statements repeat the previous analysis with 10 million observations from the simulated data. In order to fit this model in a small amount of time, a PERFORMANCE statement is added requesting that the analysis be done in distributed mode. Note that in order to run PROC HPGENSELECT in distributed node, it is necessary to license SAS High-Performance Statistics.

```
proc hpgenselect data=sgf2013(obs=10000000);
    class c:;
    model yPoisson = x: c: / dist=poisson;
    selection method=stepwise(choose=sbc);
    performance nodes     = all
```

```
                  installoc = "/opt/TKGrid"
                  gridhost  = "hpa.sas.com"
                  details;
run;
```

The PERFORMANCE statement specifies three pieces of information that are needed to execute this code in distributed mode. The GRIDHOST= and INSTALLLOC= options name the appliance and the install location of the SAS grid components on the appliance. If you intend to run several analyses in distributed mode, you can instead use SAS OPTION statements to set these values as environment variables that do not need to be specified in each procedure invocation. The NODES=ALL option in the PERFORMANCE statement requests that all the worker nodes on the appliance be used.

In this case, the data reside on the SAS client machine and are sent to the appliance and distributed to the worker nodes when PROC HPGENSELECT executes.

The "Performance Information" table in Figure 5 indicates that the computations were performed in a distributed environment with eight worker nodes, each of which uses 24 threads.

Figure 5 Distributed Mode Execution

The HPGENSELECT Procedure

Performance Information

Host Node	hpa.sas.com
Install Location	/opt/TKGrid
Execution Mode	Distributed
Grid Mode	Symmetric
Number of Compute Nodes	8
Number of Threads per Node	24

Model Information

Data Source	LOCALLIB.SGF2013_SIM
Response Variable	yPoisson
Class Parameterization	GLM
Distribution	Poisson
Link Function	Log
Optimization Technique	Newton-Raphson with Ridging

Selection Information

Selection Method	Stepwise
Select Criterion	Significance Level
Stop Criterion	Significance Level
Choose Criterion	SBC
Effect Hierarchy Enforced	None
Entry Significance Level (SLE)	0.05
Stay Significance Level (SLS)	0.05
Stop Horizon	1

Number of Observations Read	10000000
Number of Observations Used	10000000

Another indication of distributed execution is the following message, which is issued by all HPA procedures in the SAS log:

```
NOTE: The HPGENSELECT procedure is executing in the distributed
      computing environment with 8 worker nodes.
```

Figure 6 shows the "Selection Summary" table and the "Selected Effects" table. You see that this model does not contain any of the nuisance regressors, and it includes all the true effects except for **xTiny**.

Figure 6 Distributed Mode Execution

The HPGENSELECT Procedure

Selection Summary

Step	Effect Entered	Number Effects In	SBC	p Value
0	Intercept	1	43300442.4	.
1	cIn5	2	41122656.9	<.0001
2	xIn20	3	39648012.4	<.0001
3	cIn4	4	38236327.1	<.0001
4	xIn19	5	36907026.9	<.0001
5	xIn18	6	35713794.7	<.0001
6	xIn17	7	34641871.6	<.0001
7	xIn16	8	33687835.4	<.0001
8	xIn15	9	32849812.1	<.0001
9	xIn14	10	32117657.2	<.0001
10	xIn13	11	31491617.5	<.0001
11	xIn12	12	30954908.4	<.0001
12	cIn3	13	30445010.2	<.0001
13	xIn11	14	29989138.2	<.0001
14	xIn10	15	29613904.0	<.0001
15	xIn9	16	29311193.4	<.0001
16	xIn8	17	29073206.5	<.0001
17	xIn7	18	28887699.6	<.0001
18	xIn6	19	28752983.6	<.0001
19	cIn2	20	28632981.0	<.0001
20	xIn5	21	28538988.2	<.0001
21	xIn4	22	28478692.9	<.0001
22	xIn3	23	28444397.3	<.0001
23	xIn2	24	28429066.4	<.0001
24	cIn1	25	28417503.0	<.0001
25	xIn1	26	28413785.9	<.0001
26	xSubtle	27	28413770.1*	<.0001
27	xTiny	28	28413773.8	0.0004
28	xOut71	29	28413781.6	0.0039
29	xOut61	30	28413791.0	0.0098
30	xOut16	31	28413801.4	0.0167
31	xOut17	32	28413812.1	0.0198
32	xOut64	33	28413823.5	0.0294
33	xOut25	34	28413835.5	0.0427

* Optimal Value of Criterion

Selected Effects: Intercept xIn1 xIn2 xIn3 xIn4 xIn5 xIn6 xIn7 xIn8 xIn9 xIn10 xIn11 xIn12 xIn13 xIn14 xIn15 xIn16 xIn17 xIn18 xIn19 xIn20 xSubtle cIn1 cIn2 cIn3 cIn4 cIn5

Figure 7 shows the "Timing Table" that is produced when you specify the DETAILS option in the PERFORMANCE statement. You see that it took about two minutes to distribute the data to the appliance. The total run time for this analysis is about seven minutes, which is substantially faster than if this analysis had been done in single-machine mode.

Figure 7 Timing

Procedure Task Timing

Task	Seconds	Percent
Distributing Data	125.68	28.54%
Reading and Levelizing Data	62.58	14.21%
Candidate evaluation	142.91	32.45%
Candidate model fit	105.90	24.05%
Final model fit	3.32	0.75%

Now, suppose that you want to repeat this analysis with all 50 million observations in sgf2013. In practice, such large data sets are often stored in a distributed database or in HDFS on a cluster where Hadoop is installed. To illustrate how you can run the analysis alongside a database, the entire data set sgf1013 was preloaded onto a Greenplum appliance. The following statements show how you can access these data and run PROC HPGENSELECT in distributed mode alongside the database:

```
options set=GRIDHOST="hpa.sas.com";
options set=GRIDINSTALLLOC="/opt/TKGrid";

libname gridLib greenplm
        server   = "hpa.sas.com"
        user     = XXXXX
        password = YYYYY
        database = myDB;

proc hpgenselect data=gridLib.sgf2013;
    class c:;
    model yPoisson = x: c: / dist=poisson;
    selection method=stepwise(choose=sbc);
    performance details;
run;
```

Figure 8 Distributed Mode Execution

The HPGENSELECT Procedure

Performance Information

Host Node	hpa.sas.com
Execution Mode	Distributed
Grid Mode	Symmetric
Number of Compute Nodes	8
Number of Threads per Node	24

Figure 8 *continued*

Model Information

Data Source	GRIDLIB.SGF2013_SIM
Response Variable	ypoisson
Class Parameterization	GLM
Distribution	Poisson
Link Function	Log
Optimization Technique	Newton-Raphson with Ridging

Selection Information

Selection Method	Stepwise
Select Criterion	Significance Level
Stop Criterion	Significance Level
Choose Criterion	SBC
Effect Hierarchy Enforced	None
Entry Significance Level (SLE)	0.05
Stay Significance Level (SLS)	0.05
Stop Horizon	1

Number of Observations Read	50000000
Number of Observations Used	50000000

Figure 9 shows the "Selection Summary" table and the "Selected Effects" table. You see that this model includes all the true effects (even **xTiny**). However, the final model that is chosen based on the SBC includes two nuisance effects, even though all the true effects are selected before any of the nuisance effects.

Figure 9 Alongside Greenplum Execution

The HPGENSELECT Procedure

Selection Summary

Step	Effect Entered	Number Effects In	SBC	p Value
0	Intercept	1	216534570	.
1	cin5	2	205646118	<.0001
2	xin20	3	198263366	<.0001
3	cin4	4	191205816	<.0001
4	xin19	5	184546950	<.0001
5	xin18	6	178576949	<.0001
6	xin17	7	173217524	<.0001
7	xin16	8	168444762	<.0001
8	xin15	9	164255968	<.0001
9	xin14	10	160593994	<.0001
10	xin13	11	157450578	<.0001
11	xin12	12	154763299	<.0001
12	cin3	13	152223999	<.0001
13	xin11	14	149945041	<.0001
14	xin10	15	148063319	<.0001
15	xin9	16	146551826	<.0001
16	xin8	17	145358357	<.0001
17	xin7	18	144437793	<.0001
18	xin6	19	143762116	<.0001
19	cin2	20	143160713	<.0001
20	xin5	21	142689381	<.0001
21	xin4	22	142389319	<.0001
22	xin3	23	142218863	<.0001
23	xin2	24	142141849	<.0001
24	cin1	25	142085236	<.0001
25	xin1	26	142066707	<.0001
26	xsubtle	27	142066611	<.0001
27	xtiny	28	142066585	<.0001
28	xout71	29	142066571	<.0001
29	xout61	30	142066570*	<.0001
30	xout16	31	142066572	0.0001
31	xout51	32	142066577	0.0003
32	cout4	33	142066629	0.0011
33	xout47	34	142066638	0.0029
34	xout2	35	142066648	0.0054
35	xout25	36	142066658	0.0056
36	xout3	37	142066668	0.0057
37	xout78	38	142066679	0.0079
38	xout62	39	142066690	0.0100
39	xout45	40	142066701	0.0102
40	xout42	41	142066713	0.0106
41	xout76	42	142066724	0.0128
42	xout79	43	142066736	0.0139
43	xout22	44	142066748	0.0183
44	xout55	45	142066761	0.0288
45	xout23	46	142066774	0.0402
46	xout59	47	142066788	0.0454
47	xout74	48	142066802	0.0491

* Optimal Value of Criterion

Figure 9 *continued*

```
Selected Effects: Intercept xin1 xin2 xin3 xin4 xin5 xin6 xin7 xin8 xin9 xin10
                  xin11 xin12 xin13 xin14 xin15 xin16 xin17 xin18 xin19 xin20
                  xout61 xout71 xsubtle xtiny cin1 cin2 cin3 cin4 cin5
```

Figure 10 shows the timing table. Because the data are already on the appliance and PROC HPGENSELECT runs alongside the database, loading the data from the database into memory takes only a small part (less than 3%) of the overall time of about 21 minutes needed to complete this analysis.

Figure 10 Timing for Alongside Greenplum Execution

Procedure Task Timing

Task	Seconds	Percent
Distributing Data	0.34	0.03%
Reading and Levelizing Data	33.22	2.69%
Candidate evaluation	638.31	51.62%
Candidate model fit	551.57	44.60%
Final model fit	13.21	1.07%

SUMMARY

High-performance statistical modeling procedures, a new generation of SAS software for statistical modeling, are being developed based on design principles for high-performance computing with ever-increasing volumes of data. The principal goal for these procedures is building powerful predictive models. On single machines, high-performance statistical modeling procedures achieve scalability by exploiting all the cores on the machine. In distributed computing environments, these procedures exploit parallel access to the data, along with all the cores and the huge amounts of memory that are available.

REFERENCES

Cohen, R. (2002), "SAS Meets Big Iron: High Performance Computing in SAS Analytical Procedures," in *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc.

APPENDIX

```
%let nObs      = 50000000;
%let nContIn   = 20;
%let nContOut  = 80;
%let nClassIn  = 5;
%let nClassOut = 5;
%let maxLevs   = 5;

data sgf2013;
    array xIn{&nContIn};
    array xOut{&nContOut};
    array cIn{&nClassIn};
    array cOut{&nClassOut};

    drop i j sign nLevs xBeta expXBeta logitXBeta;
```

```
      do i=1 to &nObs;
         sign  = -1;
         xBeta = 0;
         do j=1 to dim(xIn);
            xIn{j} = ranuni(1);
            xBeta  = xBeta + j*sign*xIn{j};
            sign   = -sign;
         end;
         do j=1 to dim(xOut);
            xOut{j} = ranuni(1);
         end;

         xSubtle = ranuni(1);
         xTiny   = ranuni(1);

         xBeta  = xBeta + 0.1*xSubtle + 0.05*xTiny;

         do j=1 to dim(cIn);
            nLevs  = min(1+j,&maxlevs);
            cIn{j} = 1+int(ranuni(1)*nLevs);
            xBeta  = xBeta + j*sign*(cIn{j}-nLevs/2);
            sign   = -sign;
         end;

         do j=1 to dim(cOut);
            nLevs  =  min(1+j,&maxlevs);
            cOut{j} = 1+int(ranuni(1)*nLevs);
         end;

         expXBeta = exp(xBeta/20);
         yPoisson = ranpoi(1,expXBeta);
         yNormal  = xBeta + 10*rannor(1);

         logitXBeta    =  expXBeta/(1+expXBeta);

         if ranuni(1)  < logitXBeta   then yBinary = 0;
                                      else yBinary = 1;

        output;
      end;
   run;
```

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Robert Cohen
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
robert.cohen@sas.com

Robert Rodriguez
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
bob.rodriguez@sas.com