Paper CC04

Big Data – Step towards High Performance Analytics

Dhiraj Dabhi, InVentiv Health Clinical, Mumbai, India

ABSTRACT

The objective of Phase IV (post-marketing) trial is to test the safety/efficacy of the treatment on larger population. Such trials generally yield massive data. Even some Phase II/III trials which have large number of enrolled subjects or have larger study duration contain huge databases. It has been observed that SAS® programmers experience problems while working with large SAS datasets. Even basic SAS programming steps become difficult to run on such huge datasets. The problems include high compilation and execution time, SAS I/O errors, memory/disk full, unexpected errors, session time out and so on. Also as the complexity level of study derivations increases, working with huge data becomes more difficult and time consuming and inefficient programming only adds more to it. It becomes more difficult when one needs to validate such a huge data. In such scenarios, it is challenging for the programmer to work around with ever-increasing volumes of data without having to spend hours of pre-processing and preparing those data, and to be able to embed the results of the analytics. This paper provides basic details of SAS High-Performance Analytics (SAS HPA) which can help enormously in reducing the compilation and execution time. The paper also discusses some useful tips and techniques that the Programmer and Validator can use in SAS to improve efficiency while working with huge datasets.

INTRODUCTION

Clinical trials are conducted in various phases to test the safety and efficacy of drug. In early clinical development stage, small numbers of subjects are enrolled in the trial to assess the safety/tolerability of the drug. If the objective of the trial is met, it further moves to next phases. The overall sample size is then increased to collect more information on a diverse population to assess the safety/efficacy aspects of the drug. In some clinical trial designs, the volume of the data plays vital role. As the population size increases, the time taken to handle and analyze the data becomes more complex.

A traditional SAS/STAT procedure provides various procedures to analyze the data; however sometimes using the basic procedures is not enough. In this paper, we will discuss

A) The basics of SAS HPA procedures (Single and Distributed mode) and its benefits in a single machine mode. These fundamentals are discussed through the basic syntax of PROC HPSUMMARY and its comparison with the traditional SAS PROC SUMMARY procedure.
B) Tips for programmers while using BASE SAS when dealing with massive data.
C) Other useful techniques.

SAMPLE DATASET USED IN EXAMPLES:

Consider a clinical trial dataset, DM, created by appending demographic information of a single compound from several studies. This appended DM dataset has around 10 million observations and 16 variables.

	STUDYID	SUBJID	SUBJIDN	SEXN	SEXC	AGE	AGEU	HEIGHT	HEIGHTU	WEIGHT	WEIGHTU	TRTN	TRT	RACEN	RACE	I
1	X000_10000	000000000250	250	1	Male	79	years	152	cm	69	kg	3	Treatment 3 - 4 mg	4	Other	1
2	X000_10000	000000000500	500	2	Female	57	years	124	cm	55	kg	3	Treatment 3 - 4 mg	3	Black	2
3	X000_10000	000000000750	750	2	Female	34	years	197	cm	71	kg	1	Treatment 1 - 20 mg	3	Black	3
4	X000_10000	000000001000	1000	2	Female	42	years	176	cm	79	kg	2	Treatment 2 - 30 mg	2	White	4
5	X000_10000	000000001250	1250	1	Male	73	years	171	cm	92	kg	2	Treatment 2 - 30 mg	2	White	5
6	X000_10000	000000001500	1500	2	Female	56	years	195	cm	116	kg	2	Treatment 2 - 30 mg	2	White	6
7	X000_10000	000000001750	1750	1	Male	54	years	175	cm	62	kg	1	Treatment 1 - 20 mg	4	Other	7
8	X000_10000	000000002000	2000	1	Male	77	years	193	cm	90	kg	1	Treatment 1 - 20 mg	1	Asian	8
9	X000_10000	000000002250	2250	2	Female	52	years	135	cm	97	kg	2	Treatment 2 - 30 mg	1	Asian	9
10	X000_10000	000000002500	2500	1	Male	40	years	166	cm	102	kg	2	Treatment 2 - 30 mg	1	Asian	10
99999995	X000_5037	024999998750	24999998750	1	Male	74	years	176	cm	62	kg	1	Treatment 1 - 20 mg	3	Black	99999995
99999996	X000_5037	024999999000	24999999000	1	Male	71	years	136	cm	110	kg	2	Treatment 2 - 30 mg	1	Asian	99999996
99999997	X000_5037	024999999250	24999999250	1	Male	73	years	155	cm	80	kg	2	Treatment 2 - 30 mg	3	Black	99999997
99999998	X000_5037	024999999500	24999999500	1	Male	77	years	184	cm	110	kg	1	Treatment 1 - 20 mg	4	Other	99999998
99999999	X000_5037	024999999750	24999999750	1	Male	66	years	198	cm	76	kg	3	Treatment 3 - 4 mg	1	Asian	99999999
100000000	X000_5037	025000000000	25000000000	1	Male	79	years	187	cm	96	kg	2	Treatment 2 - 30 mg	3	Black	100000000

PROC CONTENTS output of DM dataset:

The SAS System

The CONTENTS Procedure

Data Set Name	WORK.DM	Observations	100000000
Member Type	DATA	Variables	16
Engine	V9	Indexes	0
Created	Thu, Sep 04, 2014 05:00:49 PM	Observation Length	144
Last Modified	Thu, Sep 04, 2014 05:00:49 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	12288
Number of Data Set Pages	1176471
First Data Page	1
Max Obs per Page	85
Obs in First Data Page	67
Number of Data Set Repairs	0
Filename	E:\SASWork_TD20376\dm.sas7bdat
Release Created	9.0202M3
Host Created	W32_SRV08

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
7	AGE	Num	8
8	AGEU	Char	5
9	HEIGHT	Num	8
10	HEIGHTU	Char	2
1	I	Num	8
16	RACE	Char	5
15	RACEN	Num	8
6	SEXC	Char	6
5	SEXN	Num	8
2	STUDYID	Char	20
3	SUBJID	Char	20
4	SUBJIDN	Num	8
14	TRT	Char	19
13	TRTN	Num	8
11	WEIGHT	Num	8
12	WEIGHTU	Char	2

A. BASICS OF HPA PROCEDURES

In order to deal with the challenges of massive data, SAS has introduced HPA procedures that leverage huge data to make faster decision making. Beginning SAS 9.4, the HPA procedures are integrated along with their associated MVA products. These procedures run in 2 different modes:

1) Single-Machine mode – In this mode the HPA procedures work on a PC, a Server and a Grid but will only uses the cores of the machine assigned to execute it.
2) Distributed mode – A cluster of nodes with SAS High-Performance Analytics Infrastructure which makes the HPA procedures very powerful. In this mode the data will be loaded into the memory distributed across multiple nodes. The execution involves the cores of all the multiple nodes.

SAS HPA procedures are integrated and work on 3 main components:

1) SAS Grid Computing – shared environment for large jobs and support a growing number of users efficiently.
2) SAS In-Database – executes logic in the database itself. It uses a MPP (Massive Parallel Processing) for faster execution of tasks.
3) SAS In-Memory Analytics – eliminates need for disk based processing for faster analysis. Divides processes into pieces with computation distributed across nodes. SAS In-Memory Analytics had a new addition announced in Dec 2011 in the form SAS High Performance Analytics. The HPA procedure enormously reduces the processing time of the model.

SAS HPA procedure can execute in symmetric multiprocessing (SMP) or massively parallel processing (MPP) mode. In SMP mode the procedure supports multithreading on client machine. The procedure uses the number of CPUs on the machine to determine the number of concurrent threads. In MPP mode the analysis occurs on several nodes of distributed computing environment. In distributed mode the HPA performs the analytics on an appliance that consists of cluster of nodes. This appliance can be:

1) a database management system (DBMS) appliance on which the SAS High-Performance Analytics infrastructure is also installed.
2) a cluster of nodes that have the SAS High-Performance Analytics infrastructure installed but no DBMS software installed

Distributed (or MPP) mode has several variations

1) **Client-data (or local-data) mode:** The input data for the analytic task are not stored on the appliance or cluster but are distributed to the distributed computing environment by the SAS High-Performance Analytics infrastructure when the procedure runs.
2) **Alongside-the-database mode:** The data are stored in the distributed database and are read from the DBMS in parallel into a high-performance analytical procedure that runs on the database appliance.
   In this mode the data is loaded in a distributed environment using PROC HPDS2, DATASTEP, PROC APPEND or SQL pass-through.
3) **Alongside-HDFS mode:** The data are stored in the Hadloop Distributed File System (HDFS) and are read in parallel from the HDFS
4) **Alongside-LASR mode:** The data are loaded from a SAS LASR Analytic Server that runs on the appliance.

DETERMINING SINGLE-MACHINE MODE OR DISTRIBUTED MODE:

HPA procedures use the following rules to determine whether they run in single or distributed mode:

1) If a grid host (DNS) is not specified, the analysis is carried out in single-machine mode on the client machine that runs the SAS session.
2) If a grid host is specified, the behavior depends on whether the execution is alongside the database or alongside HDFS. If the data are local to the client, you need to use the NODES= option in the PERFORMANCE statement to specify the number of nodes on the appliance or cluster that you want to engage in the analysis. If the procedure executes alongside the database or alongside HDFS, you do not need to specify the NODES= option.

LIST OF HP PROCEDURES INCLUDED IN THE SAS HPA SERVER:

SAS High Performance Statistics	SAS High Performance Econometrics	SAS High Performance Optimization	SAS High Performance Data Mining	SAS High Performance Text Mining	SAS High Performance Forecasting
HPLOGISTIC HPREG HPNLIN HPMIXED HPNLMOD HPSPLIT HPGENSELECT HPCORR	HPCOUNTREG HPSEVERITY HPQLIM	HPSLO Select features in OPTMILP OPTLP OPTMODEL	HPREDUCE HPNEURAL HPFOREST HP4SCORE HPDECIDE HPSVM	HPTMINE HPTMSCORE	HPFORECAST
Common set includes: HPSUMMARY, HPSAMPLE, HPDS2, HPDMDB, HPIMPUTE, HPBIN, HPCORR					

LIST OF HP PROCEDURES BY THE DATA ANALYSIS PURPOSE:

Data Preparation	Data Exploration/Transform	Analytics and Modeling	
HPDS2	HPSUMMARY	HPSUMMARY	HPSVM
HPDMDB	HPDMDB	HPLOGISTIC	HP4SCORE
HPSAMPLE	HPSAMPLE	HPREG	HPDECIDE
	HPREDUCE	HPLMIXED	HPCORR
	HPIMPUTE	HPREDUCE	HPQLIM
	HPBIN	HPNEURAL	HPNLMOD
	HPSPLIT	HPNLIN	HPGENSELECT
		HPDS2	HPSPLIT
		HPCOUNTREG	HPTMINE
		HPSEVERITY	HPTMSCORE
		HPFOREST	HPFORECAST

HPSUMMARY PROCEDURE

HPSUMMARY procedure computes the basic descriptive statistics of the variables. It can run on either single or distributed machine mode. The HPSUMMARY procedure provides the similar functionality and syntax as traditional SAS SUMMARY procedure. In general, PROC HPSUMMARY does the following:

- Calculates descriptive statistics based on moments
- Calculates and estimates quantiles, which includes the median
- Calculates CI for the mean
- Identifies extreme values
- Performs *t* test

Syntax:

PROC HPSUMMARY <*options*> <*statistics-keywords*>;
CLASS *variables* </ *options*>;

FREQ *variable*;
OUTPUT <**OUT** = *SAS-dataset*> <*output-statistic-specifications*> </ **AUTONAME**>;
PERFORMANCE *performance-options*;
TYPES *requests*;
VAR *variables* </ **WEIGHT**=*weight-variable*>;
WAYS *list*;
WEIGHT *variable*;
RUN;

Note: We can also use ATTRIB, FORMAT, LABEL and WHERE statements and any global statements.

COMPARISON BETWEEN HPSUMMARY AND SUMMARY PROCEDURE

In most of the cases the analysis performed with HPA procedure is similar to that performed with traditional SAS/STAT procedures; however with HPA procedures the performance gain can be seen with increase in the number of CPUs
In the below example we have PROC SUMMARY and PROC HPSUMMARY code run on a single machine mode with multithreading enabled with number of threads = 4.

<table>
<tr><th>PROC HPSUMMARY</th><th>PROC SUMMARY</th></tr>
<tr><td><u>Code:</u>


```
PROC HPSUMMARY data = dm;
    CLASS trt age race sexc;
    VAR height weight;
    TYPES () trt*age trt*race trt*sexc;
    OUTPUT OUT = demog_stat2 n=n mean=mean
       std=std median=median min=min max=max;
RUN;
```

</td><td><u>Code:</u>


```
PROC SUMMARY data = dm;
    CLASS trt age race sexc;
    VAR height weight;
    TYPES () trt*age trt*race trt*sexc;
    OUTPUT OUT = demog_stat2 n=n mean=mean
       std=std median=median min=min max=max;
RUN;
```

</td></tr>
<tr><td colspan="2">In Single Machine Mode</td></tr>
<tr><td><u>Log:</u>


```
206  proc hpsummary data = dm;
207      class trt age race sexc;
208      var height weight;
209      types () trt*age trt*race trt*sexc;
210
211      output out = demog_stat2 n=n mean=mean std=std median=median min=min max=max;
212  run;

NOTE: Writing HTML Body file: sashtml2.htm
NOTE: The HPSUMMARY procedure is executing in single-machine mode.
NOTE: There were 100000000 observations read from the data set WORK.DM.
NOTE: The data set WORK.DEMOG_STAT2 has 169 observations and 12 variables.
NOTE: PROCEDURE HPSUMMARY used (Total process time):
      real time           4:46.24
      user cpu time       3:57.57
      system cpu time     9.20 seconds
      memory              11083.29k
      OS Memory           27164.00k
      Timestamp           09/04/2014 09:53:21 PM
      Step Count                        12  Switch Count  0
```

</td><td><u>Log:</u>


```
213  proc summary data = dm;
214      class trt age race sexc;
215      var height weight;
216      types () trt*age trt*race trt*sexc;
217
218      output out = demog_stat1 n=n mean=mean std=std median=median min=min max=max;
219  run;

NOTE: There were 100000000 observations read from the data set WORK.DM.
NOTE: The data set WORK.DEMOG_STAT1 has 169 observations and 12 variables.
NOTE: PROCEDURE SUMMARY used (Total process time):
      real time           5:56.97
      user cpu time       4:21.53
      system cpu time     10.28 seconds
      memory              7778.31k
      OS Memory           28180.00k
      Timestamp           09/04/2014 10:14:13 PM
      Step Count                        13  Switch Count  0
```

</td></tr>
</table>

From the above result we can see the slight increase in performance of the HPA procedure. This is because most of the SAS modeling procedures support multi-threading in single machine mode. Both SUMMARY and HPSUMMARY use the multithreading functionality however HPA procedures show better performance. In HP procedures in order to modify the number of threads we can use PERFORMANCE statement in HPSUMMARY e.g., PERFORMANCE NTHREAD=10.

In the above single machine mode example although the performance increase is not significant, the actual benefits of HPA are visible when you work on distributed machine with multiple nodes (MPP mode). In certain scenarios depending on the environment in which we are running HPA procedures can run 15 to 20 times faster than the traditional SAS procedures.

ADVANTAGES OF HPA PROCEDURES

- Enables the users to run on a single machine mode where SAS is installed.
- Enables the users to run on distributed mode to distribute data and computation on different cluster of machines
- HPA procedures are multithreaded and can exploit all the cores available, whereas not all MVA procedures are multithreaded.
- In order to leverage the benefits of HPA procedures, in SAS 9.4 the procedures in the SAS HPA products are included with the associated MVA product, and we can run these procedures in single-machine mode without licensing the high-performance product.

DISADVANTAGES OF HPA PROCEDURES

- In order to enable the full functionality of HPA procedures organization need to spend extra cost to setup distributed grid environment.
- As of today we have only few HPA procedures available.

B. PROGRAMMING TIPS WHILE DEALING WITH MASSIVE DATA

While working with massive data along with the HPA benefits the SAS programmer must also follow the best practices. Below are the few tips for a programmer to follow while working with massive data.

PROBLEM 1: SIZE OF DATA

SOLUTION: It is imperative to reduce the size of the data for faster processing. In case of massive data, it is extremely important to create a dataset with reduced size to overcome few of the memory constraints. A simplest way to do this is to delete the unwanted observations and variables. It is also recommended to look at the length and attribute of the desired variables which may have major impact on the size of the datasets.

a. EFFICIENT PROGRAMMING

Various base SAS statements are available to delete the unwanted observation. The major used options are Sub-setting IF, WHERE and CASE condition. However it is preferable to use WHERE statement instead of IF statement as WHERE condition is applied on data before they enter Program Data Vector (PDV) and in case of 'IF' it is applied after data comes out from PDV.

EXAMPLE

In Demographic tables, there is requirement to analyze data for male subjects. The first step would create a reduced size data which can be achieved through PROC SQL and efficient data steps as described below.

DATASET STATEMENT	PROC SQL STATEMENT

`DATA DM1;`
`  SET DM;`
`  WHERE SEXN = 1;`
`RUN;`
	`PROC SQL;`
	`  CREATE TABLE DM1 AS`
	`  SELECT * FROM DM`
	`  WHERE SEXN = 1;`
	`QUIT;`

b. COMPRESS DATASETS:

Using the COMPRESS=YES dataset option reduces the size of data by around 10%-15% of its original data size. It is recommended that you do not compress the intermediate datasets.

c. DROP UNWANTED VARIABLES

Size of the data is directly proportional to the numbers of variables present in the data. This results in increased processing time. If the user does not wish to retain all the variables from the dataset, the unwanted variables can be dropped using the available dataset options like DROP= and KEEP= OR using DROP or KEEP statement. A suitable choice between KEEP and DROP should be made. In the above example, variable SEX is not required in the final dataset and hence DROP statement can be used rather than using KEEP statement.

EXAMPLE

```
DATA DM2;
      SET DM;
    ** Drop unwanted variable I from DM dataset;
      DROP I;
RUN;
```

d. VARIABLE LENGTHS

Reducing the length of the variables also plays an important role in reducing the overall size of the data. Suppose we have to create TRTC and DOSE variables from TRT variable as shown below.

In the example, the newly created variable TRTC and DOSE in the DM3 dataset has default length of 200 and 8 respectively. Basic programming will help to reduce the length of observation as shown below.

EXAMPLE

```
DATA DM3;
    SET DM;
    ** Set length of variables before creating;
    LENGTH trtc $15 dose 3;
    trtc = substr(trt, 1, 12);
    dose = input(substr(trt, 15, 2), best.);
RUN;
```

e. DELETE UNWANTED DATASETS

In SAS, every new data is saved in WORK library. These datasets will remain in memory until SAS session ends. The basic rule is to delete the datasets if not required for further processing. These temporary datasets can be deleted using either PROC DATASETS or PROC SQL. Deleting temporary datasets will free up lot of space which can be used by SAS for processing program or storing other datasets.

In the above sample example DM1, DM2 and DM3 is created. If we do not want these dataset for further processing it is advisable to delete the dataset as shown below.

EXAMPLE

PROC DATASETS	PROC SQL

`PROC DATASETS;`
`  DELETE DM1 DM2 DM3;`
`RUN;`
	`PROC SQL;`
	`  DROP TABLE DM1, DM2, DM3;`
	`QUIT;`

PROBLEM 2: SORTING LARGE DATASETS

SOLUTION: Sorting the huge data is always a challenge. While sorting the massive data, the major issue arises in the amount of disk space required. Sorting always takes 3-4 times more space than the actual data size, so sort the data, ONLY if required.

The most common method is to divide the data into small subset and then sort and append the data after sorting. This will help in case you have limited disk space. Another method is to adjust the SORTSIZE and MEMSIZE options to increase the amount of memory that SAS use for sorting.
You can also use the TAGSORT option to sort a large data set. The TAGSORT option stores only the BY variables and the observation numbers in temporary files. The BY variables and the observation numbers are called tags. At the completion of the sorting process, PROC SORT uses the tags to retrieve records from the input data set in sorted order. When the total length of the BY variables is small compared to the record length, TAGSORT reduces temporary disk usage considerably because sorting just the BY variables means sorting much less data. However, processing time is usually higher than a regular sort because TAGSORT increases CPU time and I/O usage in order to save memory and disk space.

EXAMPLE

```
PROC SORT DATA = <dataset-name> TAGSORT;
    BY variable-names;
RUN;
```

PROBLEM 3: MERGING THE DATASET

SOLUTION: SQL vs. Merge and FORMAT

a. **PROC SQL vs. Merge Statement**
   - The efficiency of the 2 methods depends on the type of data and the user requirements.
   - The general consideration is that, if we have to do a simple merge/inner join, data step will be more efficient than SQL code.
   - When data sets are large and unsorted, SQL INNER join will be efficient. In most cases, data step match merge generally efficient than PROC SQL OUTER JOIN.

b. **CREATING FORMAT**

This will be helpful where we need to merge the dictionary (MEDDRA or WHODRUG) information with the concomitant or medical history datasets. In the scenario, we will merge the dataset by CODE to get corresponding TERM without actually sorting and merging it.
For e.g., suppose we have a concomitant information dataset CT and the corresponding WHODRUG information from the WHODRUG dataset. Typically, we will sort the CT and WHODRUG datasets by CODE variable and then merge using data-steps merge statement. Another way to merge these 2 datasets is to get the information as a format and then use the format to get the coded TERM.

EXAMPLE

```
** Creating variable FMTNAME, START and LABEL;
DATA WHOFMT (KEEP = START LABEL FMTNAME);
       SET WHODRUG;
       FMTNAME = "WHOFMT";
       START = CODE;
       LABEL = TERM;
RUN;

** Creating Format WHOFMT from dataset;
PROC FORMAT CNTLIN = WHOFMT LIBRARY = WORK;
RUN;

** Merging Format to get the final coded information;
DATA CT1;
       SET CT;
       TERM_NAME = PUT(TERM, WHOFMT.);
RUN;
```

C. OTHER USEFUL TECHNIQUES

a. SYNTAX CHECK BEFORE RUNNING DATASET:

While running your code with big data make sure your code is syntactically correct. This will help to avoid unnecessary re-runs yielding the waste of time while running the code. In order to avoid this, before running the entire program make sure you use option OBS = N to check if your code is syntactically correct and there are zero ERROR and WARNING. N indicates number of observations SAS should process. Similarly, in PROC SQL we can use INOBS= and OUTOBS= option to restrict the row for processing. Along with this we can also use FEEDBACK, NOEXEC and VALIDATE options in PROC SQL to check if the code is syntactically correct.

b. AVOID I/O ERROR

In the SAS System, SAS I/O, the most common input/output (I/O) processing, is performed on SAS files such as SAS data sets, SAS catalogs, and SAS index files. Most SAS I/O processing is sequential I/O, which means that the DATA step and SAS procedures read a data set from the beginning of the file to the end of the file. The data is read from or written to pages that are typically 4—64 K in size. With standard SAS options, each time the dataset is read/write to a file it is first stored in a cache file. A large SAS jobs will cause numerous SAS datasets to reside in the cache. However processing large datasets (>=2GB) reduces I/O throughput. Reading several distinct SAS data sets of this size in succession causes the file cache to expand greatly. Even reading a single SAS data set of this size causes the file cache to expand because the cache manager attempts to retain as much of the data as possible. A huge file cache causes a latency each time you read from or write to a large file. The latency occurs because the cache manager has to search through the cached pages of data when it is accessing the file.

The solution is to bypass the file cache. In SAS we can implement the SGIO option in the SAS configuration. The scatter-read feature reads data directly from the disk. The gather-write feature writes data directly to the disk. To activate scatter-read/gather-write file access globally in SAS, specify the SGIO system option either on the command line or in the SAS configuration file. For example, specify the SGIO system option on a command line when you invoke SAS, as follows:

c:\SAS> sas –sgio

SAS 9.1.3 SP4 and later also contain the SGIO= data set option to enable and disable SGIO processing for specific datasets.

SGIO processing (activated with the SGIO system option) provides each SAS I/O file with the ability to use scatter read/gather-write access. However, not all SAS I/O files will be opened for SGIO processing even if the SGIO option is specified when you invoke SAS. If a SAS I/O file does not meet the required criteria (https://support.sas.com/resources/papers/IOthru**SGIO**.pdf), the SGIO processing remains inactive for that file even if you specify the SGIO system option or data set option. If a file is not using SGIO processing, no warning message appears in the SAS log. To obtain maximum throughput, use the SGIO feature apply the BUFSIZE= and the BUFNO= system options.

CONCLUSION

High Performance HPA procedures are modeling procedures are developed for high performance statistical modeling while working with massive data. In single machine mode HPA procedures takes the advantages of the multi-threading capabilities of the system. In distributed mode the HPA procedures exploits all the 3 core components while working with massive data for faster performance. Also as the syntax are similar to traditional SAS procedures its ease to learn and adapt. Furthermore along with the increasing technology advantages, the basic efficient steps help to overcome few of the memory constraints and processing time.

REFERENCE

SAS Support: Base SAS® 9.4 Procedures Guide High-Performance Procedures
http://support.sas.com/documentation/cdl/en/prochp/66704/HTML/default/viewer.htm#titlepage.htm

Robert A. Cohen and Robert N. Rodriguez, SAS Institute Inc., SAS Global Forum 2013: High-Performance Statistical Modeling
http://support.sas.com/resources/papers/proceedings13/401-2013.pdf

SAS Technical Paper: Achieving Better I/O Throughput Using SGIO in the Microsoft Windows Environment
https://support.sas.com/resources/papers/IOthru**SGIO**.pdf

John Cohen, AstraZeneca LP, Wilmington, DE, Table Lookups: Getting Started With Proc Format
http://www.lexjansen.com/nesug/nesug08/cc/cc19.pdf

CONTACT INFORMATION

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

Dhiraj Dabhi
Senior Statistical Programmer
Inventiv International Pharma Services Pvt. Ltd., India
Marwah Centre, Ground Floor, B Wing, Krishnalal Marwah Marg,
Andheri (E), Mumbai – 400 072, Maharashtra, India
T: +91 22 40957363 M: +91 9920466022
Dhiraj.Dabhi@inventivhealth.com
http://www.inVentivHealthclinical.com

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.

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