

SPSS Server Performance

Table of contents

Introduction.....	2
Server market	2
SPSS Server	2
Establishing performance baselines	3
Methodology.....	3
Traditional benchmarking.....	7
Memory and disk space usage.....	10
Overview.....	10
Future direction	11
SPSS product direction	11
Appendix A: Syntax used for tests.....	12
About SPSS Inc.	24

Introduction

Server market

Hardware vendors frequently add new technology and capabilities to their server offerings to meet the demands of today's enterprise server applications. Server machines, which increasingly have a different architecture than desktop machines, are designed specifically to facilitate large-scale processing tasks and/or several simultaneous users. Although there are many ways that a server-class machine can differ from a standard desktop product, a few notable features stand out. These include:

- Architecture
- Hyper-threading technology
- Level 2 (L2) cache

Architecture

Modern servers are now being produced with a 64-bit architecture. This allows applications to find and use more memory, which is faster than relying on disk storage for processing. When using larger files for analysis, applications can keep more of the data in memory and process data more efficiently than is possible with 32-bit processors.

Hyper-threading (HT) technology

Simultaneous multi-threading technology enables an operating system to execute different parts of a program, called threads, concurrently. Hyper-threading (HT) is Intel's implementation of simultaneous multi-threading technology, which it uses on the Intel® Pentium® 4 chipset. HT technology improves processor performance for certain types of workloads by providing functional work for execution units that would otherwise be idle. HT technology has several potential advantages:

- Improved support for multi-threaded code
- Ability to run multiple threads simultaneously
- Improved reaction and response time
- Increased number of users a server can support


Level 2 (L2) cache

Also referred to as secondary cache, L2 cache is a small amount of high-speed memory located close to and sometimes on the central processing unit (CPU). L2 cache allows for high-speed access to the system's most commonly accessed data. On newer machines, the L2 cache is most often located directly on the CPU, allowing for higher-performance memory processing.

SPSS Server

Organizations that choose to run SPSS on a server experience many benefits. For example, SPSS Server can better take advantage of system and hardware implementations, including those discussed above, that are not always available on the desktop version of SPSS. In addition, SPSS enables organizations to handle large data sources more efficiently than is possible on the desktop product. These organizations can also take advantage of features exclusive to the server version of SPSS.

SPSS Server's two-tier client/server architecture, in which the SPSS desktop or client machine is connected to a server machine, shifts data handling, preparation, and analytical tasks to the server. This prevents open transportation of data across the network to the desktop. For total data security, network administrators can create a connection that uses Secure Sockets Layer (SSL) encryption between the server and the data source.



With SPSS Server, the user gains the option to run the server in production mode by executing the SPSS Batch Facility (SPSSB). This enables the user to conduct lengthy data preparation tasks and analyses in a “lights out” mode, requiring little or no human intervention after the process begins.

Other features unique to SPSS Server allow users to:

- Sort and aggregate data inside the database prior to their retrieval for analysis
- Score new data as they are collected
- Increase performance for high-priority users
- Allocate server resources where they are needed most—rather than on a first-come, first-served basis
- Use powerful predictor selection and Naïve Bayes algorithms to identify relevant predictors in datasets that potentially have hundreds of predictors

To ensure compatibility with the current and future server market, SPSS Inc. continually implements performance improvements to SPSS Server across all platforms. In this technical report, SPSS presents test suite and traditional benchmark testing results for recent versions of SPSS Server. Other factors that influence performance, such as computer storage, hardware, and system configuration, are discussed. Additionally, SPSS offers insight into the future development plans for SPSS Server.

Establishing performance baselines

Methodology

Traditionally, “benchmarks” represent timings specific to individual commands or processes that are not truly indicative of a real-life scenario. With this in mind, the benchmarks presented in this section will show the following:

- Execution results for specific test suites, a logical grouping of processes to test a specific set of functions or capabilities, that use more real-life examples and provide statistical comparisons between releases
- Raw statistical comparisons of specific syntax across product releases, making adjustments for syntax changes and/or product enhancements

The results of these benchmark tests are shown only as SPSS Batch Facility executions because SPSSB removes some of the variability that client and server network traffic can introduce.

Test suite testing method

To determine the difference among SPSS Server releases, SPSS used scenarios thought to reasonably represent typical user implementations. Realistic scenarios provide the user with more reasonable expectations of server performance when compared to situations not indicative of real-life implementations.

To establish realistic numbers, a server machine was configured to reflect a moderate-sized implementation. SPSS avoided using a top-end machine, as it would not be a typical installation and it might skew the numbers. For installations that employ higher-end equipment, adjustments to these benchmark numbers can be made intuitively.

Test machine

The first step was to locate a machine suitable for such tests. Table 1 lists the configuration of the server machine used for the test suite benchmarks.

Server
Processor name: Intel Xeon® CPU 2.80GHz
Operating system: Microsoft® Windows® 2003 Server, Standard Edition
Type: 0
Family: F
Model: 4
Stepping: 1
Revision: D
L1 trace cache: 12Kμops
L1 data cache: 16KB
L2 cache: 1MB
L3 cache: None
Packaging: 604-pin μPGA
Enhanced Intel SpeedStep® Technology (EIST): No
MMX™ technology: Yes
Single instruction multiple data (SIMD): Yes
SIMD2: Yes
SIMD3: Yes
Enhanced halt state: Yes
Execute disable bit: Yes
Hyper-threading technology: Yes
Intel Extended Memory 64 Technology (Intel EM64T): Yes
Expected processor frequency: 2.80GHz
Reported processor frequency: 2.80GHz
Expected system bus frequency: 800MHz
Reported system bus frequency: 800MHz

Table 1: Test machine configuration

Test suites

Rather than simply time several procedures independently, the tests were structured in a more realistic manner and a group of related procedures was assembled into test suites. This grouping was meant to reflect a certain type of analysis or data processing that the SPSS user might execute in the course of a day's work. Five test suites were developed and are shown in Appendix A:

- **Data modeling:** CROSSTABS, CTABLES, NOMREG, MIXED, and REGRESSION
- **Data mining:** TREES, TWOSTEP CLUSTER, and APPLYMODEL
- **File I/O:** MATCH, ADD, SORT (numeric), SORT (string), and CASESTOVARIABLES
- **Data transformations:** Syntax to transform data using a few functions of interest (for example, SRANGE and SMOD) and some fairly complex COMPUTE statements
- **Wide files:** GET and SAVE data files as wide as 66,000 variables

Sample size

The sample size was set to $N = 5,000,000$; $p \approx 225$ variables. The SPSS (SAV) file was slightly more than 2GB.

Network traffic

The tests documented here utilize data that were local to the processor so as to remove the variability of a storage area network (SAN) or network configuration for shared disk space. It should be noted that access to any network or redundant array of independent disks (RAID)-configured storage would add overhead to the processing that is out of the control of the SPSS product. For users who want to perform their own testing, SPSS suggests that benchmarks should be performed with local storage so as to remove any variability that network or RAID processing may introduce.

Repeated trials

To help control for the chance variation of any single test run, each test suite was repeated three times. The average time in seconds is reported. Preliminary tests demonstrated that there was enough consistency for each procedure that SPSS deemed three runs were sufficient.

Test suite test results

SPSS procedures

SPSS Server and SPSSB have similar performance, allowing the user to determine the optimal execution style for the task at hand without sacrificing performance. As stated earlier, the performance benchmark tests were obtained by using SPSSB.

Data modeling						
Version	CROSSTABS	CTABLES	REGRESSION	GLM	MIXED	NOMREG
13.0.1	74	338	90	163	194	158
14.0.1	63	367	68	133	125	121
% improvement	15%	-9%	24%	19%	36%	23%
Data mining						
Version	TWOSTEP	TREES	MODEL HANDLE			
13.0.1	1299	1025	209			
14.0.1	1222	1014	176			
% improvement	6%	1%	16%			
FILE I/O						
Version	MATCH	ADD	AGGREGATE	CASES TO VARS		
13.0.1	159	96	84	15		
14.0.1	102	54	63	14		
% improvement	36%	44%	25%	9%		
SORT						
Version	STRING SORT	NUMERIC SORT				
13.0.1	554	491				
14.0.1	507	459				
% improvement	9%	6%				
TRANSFORM						
Version	SRANGE & SMOD	CFVAR & BETA	COMPLEX COMPUTE	POISSON, ETC		
13.0.1	9820	202	248	208		
14.0.1	2990	127	269	84		
% improvement	70%	37%	-8%	60%		

Table 2: Results for SPSS Server version 13.0.1 and 14.0.1 procedures, average time recorded in seconds

File processing

As Table 3 illustrates, the GET and SAVE file processing show an increase in performance that is quite substantial.

Number of variables	Version	
	13.0.1	14.0.1
GET		
100	0.01	0.04
1,000	0.06	0.04
2,000	0.13	0.07
4,000	0.28	0.15
8,000	0.54	0.32
16,000	1.10	0.60
32,000	2.41	1.22
33,000	2.78	1.30
66,000	6.44	2.36
SAVE		
100	0.02	0.02
1,000	0.10	0.09
2,000	0.21	0.16
4,000	0.44	0.31
8,000	0.87	0.60
16,000	1.79	1.22
32,000	4.05	2.13
33,000	4.37	2.14
66,000	11.10	4.53

Table 3: GET/SAVE wide files, SPSS Server version 13.0.1 and 14.0.1

Traditional benchmarking

There are instances when batch processing is required for the enterprise implementation. SPSSB allows the user to execute syntax in a stand-alone environment. The performance information provided in Table 2 reflects more real-life implementations, but there is also a need to document performance data that reflect a specific execution of commands. The traditional benchmark testing discussed below focuses on those commands that require reading and writing data.

Traditional testing method

SPSS wanted to determine the performance of SPSS Server using scenarios thought to reasonably represent some of the more common syntax commands that may be affected by reading or writing data. By separately focusing on data-intensive operations, the server user receives a comprehensive understanding of the individual commands and how they should perform.

To establish realistic numbers, SPSS configured a server machine to reflect a moderate-sized implementation. A top-end machine was avoided, as it would not be a typical installation and it might skew the numbers. For installations that employ higher-end equipment, adjustments to these benchmark numbers can be made intuitively.

Test machine

Table 4 lists the configuration of the server machine used for the traditional benchmark testing.

Server
Processor name: Intel Pentium® III CPU 500 MHz (dual processor)
Operating system: Microsoft Windows Server 2003 Standard Edition
Memory: 1024MB PC100 SDRAM
Hard drive: (2) Maxtor® 20GB, ATA 33

Table 4: Test machine configuration

Traditional comparisons

A common user scenario is to read data from a source file, collect aggregate information on the observed variables in that file (for example, minimum, maximum, mean, sum, standard deviation, count, etc.), and then match the aggregated information to the source file. This test was set up to compare the results between older SPSS releases and the recent SPSS 13.0 and 14.0 releases. In SPSS 13.0 and higher, this task is completed in a single step using the AGGREGATE command and the /MODE=ADDVARIABLES subcommand. In prior versions, it was necessary to first read the data, sort and aggregate them, write an external file of the aggregated values, and then merge these aggregated values into the active source file. To compare and contrast performance for these exact steps across four of the most recent releases of SPSS for Windows (versions 12.0.2, 13.0.1, 14.0.0, and 14.0.1), SPSS developed three command syntax files. The syntax is shown in Appendix A and summarized below:

- **MegaCompute.sps:** This is a data transformations-intensive job using LOOP—END LOOP and COMPUTE command syntax. It is intended to elicit information explaining how each version of SPSS for Windows utilized the available computing power of the test server without needing file input/output.
- **Procs.sps:** This file represents the standard user scenario. After reading the file, SPSS ran two representative procedures on the unsorted data (such as DESCRIPTIVES and CTABLES). Next, the data were sorted (i.e., SORT), aggregated to an external file (i.e., AGGREGATE), and the aggregated values were merged into their originating cases in the active file (i.e., MATCH FILES and EXECUTE).
- **Sort-13.sps:** For comparison, AGGREGATE with /MODE=ADDVARIABLES subcommand in SPSS 13.0 and and SPSS 14.0 was run on the four test versions and timings were collected.

Source data

The source data file includes 27 mixed variables (including long strings, short strings, real numbers, and integers as dates) and 772,369 cases. The SAV file is slightly more than 130MB in size.

Network traffic

The tests documented here utilize data that were local to the processor so as to remove the variability of a SAN or network configuration for shared disk space. Furthermore, the test server was physically removed from the network so that any Internet or network processing would not alter the findings. It should be noted that access to any network or RAID-configured storage will add overhead to the processing that is out of the SPSS product's control. For users who want to perform their own testing, SPSS suggests that benchmarks should be performed with local storage so as to remove any variability that the network or RAID processing may introduce.

Repeated trials

To help control for the chance variation of any single test run, each test suite was repeated five times per SPSS version with the versions tested being counterbalanced (i.e., SPSS 12.0, 13.0, 14.0, 13.0, 14.0, 12.0, 14.0, 12.0, 13.0, ..., etc.) so as to not favor any one release. The minimum, maximum, and mean times were collected and documented.

Traditional benchmark results

SPSS procedures

As Table 5 below shows, SPSS 14.0.1 outperforms previous versions significantly. Specifically, the table reports on the performance across versions for the MegaCompute.sps (MegaCompute row), the Procs.sps (DESC, CTABLES, SORT, AGGR12, and MATCH rows), and the Sort-13.sps (AGGR13 row) test jobs. Note especially the AGGR12_tot row. This row represents the total time taken to perform the procedures in the Procs.sps test file. Procs.sps represents the only way to perform this user scenario in SPSS 12.0. Compare these results to the AGGR13 row for SPSS 13.0 and 14.0 versions using the new ADDVARIABLES subcommand.

Report					
	SPSS Version	Minimum	Maximum	Mean	Std. Deviation
Mega Compute	12.0.2	94.21	94.73	94.3800	.20187
	13.0.1	101.41	101.95	101.5620	.22874
	14.0.0.260	93.92	93.98	93.9420	.01483
	14.0.1.6	97.02	98.01	97.9120	.09330
DESC	12.0.2	12.23	12.23	12.2300	.00000
	13.0.1	12.62	12.62	12.6200	.00000
	14.0.0.260	15.22	15.64	15.3640	.16072
	14.0.1.6	9.12	9.13	9.1200	.00447
CTABLES	12.0.2	20.74	20.77	20.7540	.01517
	13.0.1	15.65	15.66	15.6580	.00447
	14.0.0.260	19.48	19.70	19.6080	.10114
	14.0.1.6	11.81	11.90	11.8740	.03782
SORT (sec)	12.0.2	41.54	42.15	41.9000	.25750
	13.0.1	41.96	42.75	42.2880	.30161
	14.0.0.260	40.10	40.83	40.4400	.23738
	14.0.1.6	28.83	29.50	29.1280	.39061
AGGR12	12.0.2	7.94	7.97	7.9460	.01342
	13.0.1	7.41	7.47	7.4280	.02683
	14.0.0.260	12.83	13.21	12.9300	.15796
	14.0.1.6	8.89	8.94	8.9140	.01817
MATCH	12.0.2	29.20	29.37	29.2820	.07791
	13.0.1	30.28	30.83	30.5780	.21206
	14.0.0.260	29.13	29.59	29.3340	.18515
	14.0.1.6	18.07	18.35	18.1280	.12522
AGGR13	13.0.1	34.71	34.74	34.7300	.01225
	14.0.0.260	30.28	30.59	30.4340	.12462
	14.0.1.6	22.89	22.93	22.9080	.01843
AGGR12_tot	12.0.2	78.86	79.30	79.1280	.20290
	13.0.1	79.02	80.77	80.2940	.36143
	14.0.0.260	82.09	83.24	82.7040	.46811
	14.0.1.6	51.62	52.48	52.1660	.40961

Table 5: Results for SPSS Server version comparison

Memory and disk space usage

Overview

There are many performance issues centered on disk space and memory usage by software products. Disk space is less expensive than physical memory and is much more plentiful. As the information being analyzed by corporations grows, however, disk space usage becomes an increasingly important factor in system performance and tuning. The amount of available memory also plays an essential role in performance. For example, having more physical memory available allows for more concurrent users as well as larger file processing. Therefore, computer storage—disk or memory—is vital to overall performance.

Disk

SPSS Server uses disk space as temporary workspace for some procedures and for sorting. So as to limit the amount of disk contention, the temporary space should be analyzed in relation to the other available disk space to provide the best possible performance for the system. Optimally, the temporary storage space should be physically separated from the data and operating system disk space. SPSS Server allows the network administrator to designate the directory for users' temporary space to minimize the amount of disk contention and optimize the temporary space utilization per installation.

Network administrators should allocate disk space based on the number of users and size of datasets being accessed. Each user typically needs temporary disk space equal to approximately two times the size of the data (SAV) file in use (recommended range is from 1 to 2.5 times). A user sorting a file may need temporary space of approximately three times the size of the file. The disk space is used for temporary workspace as well as the final output of the sorted file. For example, if six concurrent users are accessing a file, assuming two might be sorting it, the maximum they'll need is 17 times its size. In practice, they will not be operating at peak usage, so 12 times the size will usually be sufficient.

Hardware

The configuration of the disk drive makes a difference in performance as well. In order to provide the fastest access, SCSI disk drives are recommended. Most server-class machines come with internal SCSI drives that are preferable to IDE-type drives. By using SCSI drives attached directly to the motherboard of the server, it's possible for the temporary files to reduce overhead and have less influence on system performance.

System configuration

In order to reduce the contention between data and temporary disk usage, it is best to keep temporary files on a separate disk spindle. If the data files and the temporary disk space are on the same spindle (partitioning a spindle to multiple drives has this effect), the reading and writing will contend for the same physical resource. Separating them if possible and placing them on a different channel or controller will improve performance.

RAID disk arrays should never be considered for temporary files. RAID will add to the overhead of the read and write operation and any striping will have a negative effect on the sequential processing done by SPSS Server. In addition, the recoverability that is provided as part of a RAID array is not something temporary files would utilize. If the system must be configured to use RAID, it is recommended that RAID0 be used for the temporary file spindle—because the performance degradation from RAID0 is minimized over the redundancy gained from RAID1. RAID disk arrays should only be considered for the data files, and using a RAID1 or RAID0 would be the best alternatives to consider. Since the SAV files are accessed sequentially for processing, any random striping would cause performance degradation for the system and should be avoided if at all possible.

CPU file compression (Windows only)

If CPU utilization is not a problem, users should compress the data directory or data files on disk (Windows only). This should further reduce the physical amount of disk I/O that may need to take place to read and write data. By reducing the physical I/O at the cost of available CPU cycles, the processing should see a performance boost if there is actual compression of the data over and above the compression the SAV files already provide.

Additional virtual memory should not be allocated arbitrarily. The more virtual memory is allocated, the more the operating system may begin to thrash. Thrashing is defined as excessive paging of the operating system to free up memory for processing. Once thrashing occurs, the processing will degrade, and it may eventually cause the computer to virtually halt processing.

SPSS configuration

SPSS Server has the ability to aid memory allocation for certain processes. The SET WORKSPACE command allows the user to set the memory allocation (workspace) by taking advantage of informational messages provided as part of some of the procedures. If memory is not a problem but disk usage is, the network administrator can set the workspace parameters higher to possibly achieve faster performance. Generally, administrators can take the expected number of concurrent users and divide that number by the amount of RAM on the server machine. For example, for an average of four concurrent users with a server computer of 1GB of RAM, the administrator can set workspace to .25GB RAM. Setting the workspace too high can cause thrashing; however, setting it too low can cause certain procedures to rely on too much temporary disk space, so it is important to carefully establish this setting.

Future direction

SPSS product direction

SPSS Inc. is dedicated to producing an enterprise-class product for its customers. SPSS Server and SPSS Batch facility provide implementation options that utilize the power of server hardware. In a continued quest for improvement, SPSS has identified two specific areas, optimization and hyper- and multi-threading, on which to concentrate in coming releases.

Optimization

SPSS will focus on optimizing the 32-bit (non-Windows platforms) and 64-bit (all platforms) versions of SPSS Server in future releases. It is important to strike a balance between speed and accuracy. To that end, the company will monitor this issue closely to ensure that accuracy is maintained and not sacrificed for the sake of speed.

- **32-bit:** SPSS will continue research to further improve 32-bit performance but will concentrate on the 64-bit platforms to parallel the hardware industry releases.
- **64-bit:** SPSS is currently researching options to optimize the software for multiple chipsets rather than restricting usage to a specific one. SPSS will continue to monitor the marketplace to determine the best possible solution for its customer base. SPSS supports multiple platforms and is dedicated to addressing platform changes as they occur.

Hyper- and multi-threading

SPSS is conducting research to determine what changes could be made to better utilize this and other hardware features. SPSS will work with hardware partners to determine how best to utilize new functionality as it is introduced. The company has identified possible opportunities that will allow it to take advantage of some of the newer hardware features, and will continue to evolve the product line to meet the market's needs.

Appendix A: Syntax used for tests

The syntax used for the SPSS procedure tests

Note: Tests were conducted in October 2005.

* **Transformations** — Exercise the transformations subsystem.

set printback none.

*spss 13.0.

*modification history.

* 04/07/05 - richardm - created.

*description.

* This job is designed to assess the performance of SPSS version 13.0.

* This job needs to be run in SPSS version 13.0, client and server,

* both through the frontend and in SPSSB.

file handle Source /name='<Your Path to SPSS Performance Data>'.
file handle Temp /name="<Your temp path>".
file handle timings /name="Temp\Compute_Timings.sav".
file handle times /name="Temp\times.dat".
file handle Data /name='Source\performance_data.sav'.

set work=20000 /messages on /printback on /mxerrs=99999 /mxwarns=99999.

sho ver.

oms select tables /if subtypes=["System Variables"] /destination format=sav numbered="Number" outfile=timings.

get file data.

recode y (else=copy).

exe. /* Force a data pass.

compute gender=trunc(uniform(2)).

* CDF.SRANGE(quant, a, b) returns the cumulative probability that a value from the

* Studentized range statistic, with the specified parameters, will be less than quant.

* CDF.SMOD(quant, a, b) returns the cumulative probability that a value from the

* Studentized maximum modulus, with the specified parameters, will be less than quant.

* (first parameter is "value", second is # samples, & third is df).

compute cdf_studentized_range=cdf.srange(y1,v1,y2).

compute cdf_studentized_max_modulus=cdf.smod(y1,v1,y2).

compute cdf_binomial=cdf.binom(y2,v8,.50).

compute cdf_gamma=cdf.gamma(min(y5,1),min(y1,y2,y3),max(y1,y2,y3)).

compute min=min(z1 to z37).

show \$vars.

exe.

show \$vars.

NUMERIC age (F4.0).

COMPUTE zage=rv.beta(2,3).

COMPUTE age=trunc(18+62*zage).

/**** Create ED ***.

/* Levels of education range from 6-24, or age-5, whichever
/* is lower, slightly weighted to high school degrees.

NUMERIC ed (F4.0).
COMPUTE ed=min(trunc(18*rv.beta(4-2*zage,3.25)+6),age-(mean(y1 to y37))).

/**** Create EMPLOY, RETIRE ***
/* Years of employment range uniformly from 0 to the minimum
/* of age and retirement age, minus the maximum of 15 and 5+ed.

NUMERIC employ (F4.0).
COMPUTE retire = trunc(rv.uniform(55,80)).
DO IF (min(age,retire)-max(15,5+ed) > 0).
COMPUTE employ=trunc(rv.uniform(0,min(age,retire)-max(15,cfvar(sd(z1 to z37),mean(z1 to z37))+ed))).
ELSE.
COMPUTE employ=0.
END IF.

show \$vars.
exe.
show \$vars.

/**** Create INCOME ***.
/* Income is dependent upon, and positively affected by, AGE
/* ED, and EMPLOY. Higher levels of education correspond to
/* higher variance in income.

DO IF (retire>age).
COMPUTE retire=0.
ELSE.
COMPUTE retire=1.
END IF.

COMPUTE income=(trunc((rv.beta(1,5)**retire)*
 (exp(.08*ed+.0008*(100-age)*
 age+.07*employ+rv.normal(0,ed/30+.05))))+9)*
 (cfvar(sd(z1 to z37),mean(z1 to z37))*
 1000*cdf.gamma(min(y5,1),min(y1,y2,y3),max(y1,y2,y3))).

show \$vars.
exe.
show \$vars.

FORMAT income (DOLLAR8.2).

/**** Create MARITAL ***.

NUMERIC marital (F4.0).
COMPUTE marital=rv.bernoulli(rv.beta(0.5+gender*(1-zage)+(1-gender)*zage,1)).

/* The number of people in the household depends upon MARITAL and AGE.

NUMERIC reside (F4.0).
COMPUTE reside=1+marital+(rv.poisson(1.5-zage)+marital)*rv.bernoulli((.6+.4*marital)*(1-zage)).

show \$vars.
exe.
show \$vars.

OMSEND.

** CALCULATE TIMES **.

```
get file timings /keep var1 value.  
sel if var1="$TIME".  
exe.
```

```
write out times /"" value "".  
exe.
```

```
data list list file=times /value (datetime40.5).  
string Procedure (a15).  
compute #mod=mod($casenum,2).  
do if #mod=0.  
compute diff=value-lag(value).  
if $casenum=2 procedure="SRANGE & SMOD".  
if $casenum=4 procedure="CFVAR & BETA".  
if $casenum=6 procedure="COMPLEX COMPUTE".  
if $casenum=8 procedure="POISSON ETC".  
end if.  
exe.
```

```
sel if not(sysmis(diff)).  
sum var procedure diff /for validlist noc not /cells sum /sta none.
```

*=== end of job ===.

* **Sort** — Exercise the Sort.

```
set printback none.  
*spss 13.0.
```

```
*modification history.  
* 04/07/05 - richardm - created.
```

```
*description.  
* This job is designed to assess the performance of SPSS version 13.0.  
* This job needs to be run in SPSS version 13.0, client and server,  
* both through the frontend and in SPSSB.
```

```
file handle Source /name='<Your Path to SPSS Performance Data>'.  
file handle Temp /name="<Your temp path>".  
file handle timings /name="Temp\Modeling_Timings.sav".  
file handle times /name="Temp\times.dat".  
file handle Data /name='Source\performance_data.sav'.
```

```
set work=20000 /messages on /printback on /mxerrs=99999 /mxwarns=99999.
```

```
sho ver.
```

```
oms select tables /if subtypes=["System Variables"] /destination format=sav numbered="Number" outfile=timings.
```

```
get file data.  
recode y (else=copy).  
exe. /* Force a data pass.
```

sub "STRING SORT".

show \$vars.
sort cases by ssn.
show \$vars.

sub "NUMERIC SORT".

get file data.
recode y (else=copy).
exe. /* Force a data pass.

show \$vars.
sort cases by y15.
show \$vars.

OMSEND.

** CALCULATE TIMES **.

get file timings /keep var1 value.
sel if var1="\$TIME".
exe.

write out times /"" value ""'.
exe.

data list list file=times /value (datetime40.5).
string Procedure (a12).
compute #mod=mod(\$casenum,2).
do if #mod=0.
compute diff=value-lag(value).
if \$casenum=2 procedure="SORT STR".
if \$casenum=4 procedure="SORT NUM".
end if.
exe.

sel if not(sysmis(diff)).
sum var procedure diff /for validlist noc not /cells sum /sta none.

new file.
erase file times.

*=== end of job ===.

* **Modeling** — Exercise the modeling.

set printback none.
*spss 13.0.

*modification history.
* 04/07/05 - richardm - created.

*description.
* This job is designed to assess the performance of SPSS version 13.0.
* This job needs to be run in SPSS version 13.0, client and server,
* both through the frontend and in SPSSB.

```
file handle Source /name='<Your Path to SPSS Performance Data>'.
file handle Temp /name='<Your temp path>'.
file handle timings /name='Temp\Modeling_Timings.sav'.
file handle times /name='Temp\times.dat'.
file handle Data /name='Source\performance_data.sav'.
```

```
set work=20000 /messages on /printback on /mxerrs=9999 /mxwarns=9999.
```

```
sho ver.
```

```
get file data.
recode y (else=copy).
exe. /* Force a data pass.
```

```
oms select tables /if subtypes=["System Variables"] /destination format=sav numbered="Number" outfile=timings.
```

```
sub "CROSSTABS".
```

```
show $vars.
show $vars. /* Ignore these first two SHOW $VARS commands.
```

```
show $vars.
cro tab y24 to y30 by y35 /cells count exp /sta chisquare.
show $vars.
```

```
sub "CTABLES".
```

```
show $vars.
ctables
  /vlabels variables=y1 v4 v3 v2 v1 display=default
  /table v4[c] > v3 [c] > v2 [c] > v1 [c][count f40.0] by y1
  /categories variables=y1 v4 v3 v2 v1 order=a key=value empty=exclude.
show $vars.
```

```
sub "REGRESSION".
```

```
show $vars.
reg /dep y /met ent z1 to z11 v20 to v35.
show $vars.
```

```
sub "GLM".
```

```
show $vars.
glm y WITH z1 z2 z3 z4 z5 z6 z7 z8 z9 z10 z11
  v20 v21 v22 v23 v24 v25 v26 v27 v28 v29 v30 v31 v32 v33 v34 v35
  /METHOD = SSTYPE(3) /INTERCEPT = INCLUDE
  /CRITERIA = ALPHA(.05)
  /DESIGN = z1 z2 z3 z4 z5 z6 z7 z8 z9 z10 z11
  v20 v21 v22 v23 v24 v25 v26 v27 v28 v29 v30 v31 v32 v33 v34 v35.
show $vars.
```

```
sub "MIXED".
```

```
show $vars.
MIXED y BY row2 y12 WITH z28 z29
  /FIXED = row2 y12 | SSTYPE(3)
  /METHOD = REML
  /PRINT = CPS SOLUTION TESTCOV
```

```

/RANDOM INTERCEPT I COVTYPE(CS).
show $vars.

sub "NOMREG".

show $vars.
NOMREG y18 (BASE=LAST ORDER=ASCENDING) BY y11 y13 y15 WITH y21 y27 y30 y32
/CRITERIA CIN(95) DELTA(0) MXITER(100) MXSTEP(5) CHKSEP(20) LCONVERGE(0)
PCONVERGE(0.000001) SINGULAR(0.00000001)
/MODEL y11 y13 y15 y21 y27 y30 y32
/STEPWISE=PIN(.05) POUT(0.1) MINEFFECT(0) RULE(SINGLE)
/INTERCEPT=INCLUDE
/PRINT=PARAMETER SUMMARY LRT CPS STEP MFI.
show $vars.

show $vars.
show $vars. /* Ignore these last two SHOW $VARS commands.

OMSEND.

** CALCULATE TIMES **.

get file timings /keep var1 value.
sel if var1="$TIME".
exe.

write out times /"" value "".
exe.

data list list file=times /value (datetime40.5).
string Procedure (a12).
compute #mod=mod($casenum,2).
do if #mod=0.
compute diff=value-lag(value).
if $casenum= 4 procedure="CROSSTABS".
if $casenum= 6 procedure="CTABLES".
if $casenum= 8 procedure="REGRESSION".
if $casenum=10 procedure="GLM".
if $casenum=12 procedure="MIXED".
if $casenum=14 procedure="NOMREG".
end if.
exe.

sel if not(sysmis(diff)).
sum var procedure diff /for validlist noc not /cells sum /sta none.

new file.
erase file times.

*=== end of job ===.

* Mining — Exercise the data mining.

set printback none.
*spss 13.0.

*modification history.
* 04/07/05 - richardm - created.

```

*description.

* This job is designed to assess the performance of SPSS version 13.0.

* This job needs to be run in SPSS version 13.0, client and server,

* both through the frontend and in SPSSB.

file handle Source /name='<Your Path to SPSS Performance Data>'.
file handle Temp /name="<You temp path>".

file handle timings /name="Temp\Mining_Timings.sav".

file handle times /name="Temp\times.dat".

file handle Data /name='Source\performance_data.sav'.
file handle treemodel /name="Temp\tree_model.xml".

set work=20000 /messages on /printback on.

set work=20000 /messages on /printback on.

sho ver.

get file data.

recode y (else=copy).

exe. /* Force a data pass.

oms select tables /if subtypes=["System Variables"] /destination format=sav numbered="Number" outfile=timings.

sub "TWOSTEP".

show \$vars.

TWOSTEP CLUSTER

/CATEGORICAL VARIABLES=y1 y2 y3 y4 y5 y6

/CONTINUOUS VARIABLES=y z11 z16 z18 z21 z22

/DISTANCE LIKELIHOOD

/NUMCLUSTERS AUTO 15 BIC

/NOSTANDARDIZE z11 z16 z18 z21 z22

/HANDLENOISE 0

/MEMALLOCATE 256

/CRITERIA INITHRESHOLD (0) MXBRANCH (8) MXLEVEL (3)

/PRINT COUNT SUMMARY.

show \$vars.

sub "TREES".

show \$vars.

TREE y19 [n] BY y17 [o] y18 [o] y20 [o] y21 [o] y22 [o] y23 [o] y24 [o]

v15 [s] v18 [s] v26 [s] v37 [s] v20 [s] v22 [s] v31 [s]

v34 [s] z37 [s]

/TREE DISPLAY=NONE

/DEPCATEGORIES USEVALUES=[VALID]

/PRINT MODELSUMMARY CLASSIFICATION RISK

/METHOD TYPE=CHAID

/GROWTHLIMIT MAXDEPTH=3 MINPARENTSIZE=50000 MINCHILDSIZE=25000

/CHAID ALPHASPLIT=0.05 ALPHAMERGE=0.01 SPLITMERGED=NO CHISQUARE=LR CONVERGE=0.001

MAXITERATIONS=100 ADJUST=BONFERRONI INTERVALS=10

/OUTFILE TRAININGMODEL=TreeModel

/INFLUENCE weight_variable.

show \$vars.

sub "MODEL HANDLE (SERVER ONLY)".

model handle name=treemodel file=treemodel.

```

compute predicted_value=applymodel(treemodel,predict).
compute terminal_node_number=applymodel(treemodel,nodeid).

show $vars.
exe.
show $vars.

OMSEND.

** CALCULATE TIMES **.

get file timings /keep var1 value.
sel if var1="$TIME".
exe.

write out times /"" value "".
exe.

data list list file=times /value (datetime40.5).
string Procedure (a12).
compute #mod=mod($casenum,2).
do if #mod=0.
compute diff=value-lag(value).
if $casenum=2 procedure="TWOSTEP".
if $casenum=4 procedure="TREES".
if $casenum=6 procedure="MODEL HANDLE".
end if.
exe.

sel if not(sysmis(diff)).
sum var procedure diff /for validlist noc not /cells sum /sta none.

new file.
erase file times.

*=== end of job ===.

* Make Match and Add Files.

file handle Source /name='<Your Path to SPSS Performance Data>'.

get file "Source/performance_data.sav".
compute id=$casenum.

save out "Source/match_data_01.sav" /keep=id v1 to v37.
save out "Source/match_data_02.sav" /keep=id z1 to z37.
save out "Source/match_data_03.sav" /keep=id y1 to y37.
save out "Source/match_data_04.sav" /keep=id cluster.
save out "Source/match_data_05.sav" /keep=id y row1 to weight_variable.

get file "Source/performance_data.sav".
compute id=$casenum.
exe.

temp.
sel if id le 1000000.
save out "Source/add_data_01.sav".

```

```
temp.  
sel if range(id,1000001,2000000).  
save out "Source/add_data_02.sav".
```

```
temp.  
sel if range(id,2000001,3000000).  
save out "Source/add_data_03.sav".
```

```
temp.  
sel if range(id,3000001,4000000).  
save out "Source/add_data_04.sav".
```

```
temp.  
sel if range(id,4000001,5000000).  
save out "Source/add_data_05.sav".
```

```
*=== end of job ===.
```

*** Make Data Files .**

```
** To create data.
```

```
new file.  
input program.  
vector v (37).  
vector z (37).  
vector y (37).  
string #a (a16).  
loop cluster = 1 to 1.  
+ leave cluster.  
+ loop row1 = 1 to 1.  
+ leave row1.  
+ loop row2 = 1 to 5.  
+ leave row2.  
+ loop col1 = 1 to 1.  
+ leave col1.  
+ loop col2 = 1 to 2.  
+ leave col2.  
+ compute y=100+normal(15).  
+ compute weight_variable=trunc(1+uniform(3)).  
+ loop #i=1 to 37.  
+ compute v(#i)=trunc(10+normal(2)).  
+ compute y(#i)=1+trunc(uniform(5)).  
+ compute #temp=normal(1).  
+ compute #a=string(#temp,F4.2).  
+ compute z(#i)=number(#a,F4.2).  
+ end loop.  
+ end case.  
+ end loop.  
+ end loop.  
+ end loop.  
+ end loop.  
+ end loop.  
end loop.  
end file.  
end input program.
```

```

string #a1 to #a9 (A1) ssn (a11).
vector a=#a1 to #a9.
loop #i=1 to 9.
compute a(#i)=string(trunc(uniform(10)),f1).
end loop.
compute ssn=concat(#a1,#a2,#a3,"-",#a4,#a5,"-",#a6,#a7,#a8,#a9).

format v1 to v37 y1 to y37 weight_variable row1 row2 col1 col2 (F2.0).
variable level y1 to y37 (ordinal).
variable level row1 row2 col1 col2 cluster (nominal).

save out "<YOUR DRIVE>\<YOUR PATH>\<YOUR FILENAME HERE.SAV>".

show n.

*=== end of job ===.

* File I/O — Exercise the file I/O subsystem.

set printback none.
*spss 13.0.

*modification history.
* 04/07/05 - richardm - changed time calculation.

*description.
* This job is designed to assess the performance of SPSS version 13.0.
* This job needs to be run in SPSS version 13.0, client and server,
* both through the frontend and in SPSSB.

file handle Source /name='<Your Path to SPSS Performance Data>'.
file handle Temp /name="<Your temp path>".
file handle timings /name="Temp\Modeling_Timings.sav".
file handle times /name="Temp\times.dat".
file handle Data /name='Source\performance_data.sav'.
file handle match1 /name="Source\match_data_01a.sav".
file handle match2 /name="Source\match_data_02a.sav".
file handle match3 /name="Source\match_data_03a.sav".
file handle match4 /name="Source\match_data_04a.sav".
file handle match5 /name="Source\match_data_05a.sav".
file handle add1 /name="Source\add_data_01.sav".
file handle add2 /name="Source\add_data_02.sav".
file handle add3 /name="Source\add_data_03.sav".
file handle add4 /name="Source\add_data_04.sav".
file handle add5 /name="Source\add_data_05.sav".

set work=20000 /messages on /printback on.

oms select tables /if subtypes=["System Variables"] /destination format=sav numbered="Number" outfile=timings.

sub "MATCH".

get file match1.
match files /file=* /file=match2 /file=match3 /file=match4 /file=match5 by id.

show $vars.
exe.
show $vars.

```

```

sub "ADD".

get file=add1.
add files file=* /file=add2 /file=add3 /file=add4 /file=add5.

show $vars.
exe.
show $vars.

get file data.

show $vars.
agg out */break=row1 row2 col1 col2 /depvar=mean(y) /v1 to v37=pin(v1 to v37,1,5).
show $vars.

sub "CASES TO VARS".

get file data /keep row1 row2 col1 col2 cluster y.
SORT CASES BY row1 row2 col1 col2 cluster.

show $vars.
CASESTOVARS /ID = row1 row2 col1 col2 /INDEX = cluster /GROUPBY = INDEX.
show $vars.

get file data.
n 1250000.
des var y /* Pass the data.

show $vars.
VARSTOCASES /MAKE trans1 FROM row1 row2 col1 col2
/INDEX=index1(3) /KEEP=col2 cluster row1 row2 col1 weight_variable z1 to z37 /NULL=KEEP.
show $vars.

OMSEND.

** CALCULATE TIMES **.

get file timings /keep var1 value.
sel if var1="$TIME".
exe.

write out times /"" value "".
exe.

data list list file=times /value (datetime40.5).
string Procedure (a12).
compute #mod=mod($casenum,2).
do if #mod=0.
compute diff=value-lag(value).
if $casenum= 2 procedure="MATCH".
if $casenum= 4 procedure="ADD".
if $casenum= 6 procedure="AGGREGATE".
if $casenum= 8 procedure="CASESTOVARS".
if $casenum=10 procedure="VARSTOCASES".
end if.
exe.

```

```
sel if not(sysmis(diff)).
sum var procedure diff /for validlist noc not /cells sum /sta none.
```

```
new file.
erase file times.
```

```
*=== end of job ===.
```

The syntax used for the batch processing tests [Second-level subhead]

* **MegaCompute** — Exercise the transformations subsystem.

```
set mxloops 1000000000.
set mess on.
```

```
new file.
input program.
numeric x y z LowZ HighZ.
string Formatted(a8).
leave LowZ HighZ.
compute LowZ = 10**300.
compute HighZ = -LowZ.
loop #OuterLoop = 1 to 6.
. compute x = trunc(unif(9.999)).
. do repeat #GenCode = 1 to 10.
.   loop #InnerLoop = 1 to 160000.
.     compute y = x*x+5.
.     compute z = y/7.
.     compute Formatted = string(z,f8.2).
.     compute LowZ = min(LowZ,z).
.     compute HighZ = max(HighZ,z).
.   end loop.
. end repeat.
end loop.
end case.
end file.
end input program.
execute.
list.
```

* **Procs.sps** – This was the standard user scenario.

```
SET WORKSPACE=768000/PRINTBACK=LISTING /MESSAGES=ON.
GET
FILE='E:\XXX\SPSSextract.sav'.
```

```
DESCRIPTIVES
```

```
VARIABLES=Alevel GCSE GNVQ PreGNVQ NVQ
/STATISTICS=MEAN STDDEV MIN MAX .
```

```
* Custom Tables.
```

```
CTABLES
```

```
/VLABELS VARIABLES=PartTime Gender DISPLAY=DEFAULT
/TABLE Gender [COUNT F40.0] BY PartTime
/CATEGORIES VARIABLES=PartTime Gender ORDER=A KEY=VALUE EMPTY=EXCLUDE.
```

```
SORT CASES BY LEA Estab.
```

```
AGGREGATE OUTFILE='E:\XXX\aggr.sav'
/PRESORTED
/BREAK=LEA Estab
```

```
/DOB_mean = MEAN(DOB) /EntryDate_mean = MEAN(EntryDate).  
MATCH FILES  
/FILE=*  
/TABLE='E:\XXX\aggr.sav'  
/BY LEA Estab.  
EXECUTE.
```

* **Sort-12** – This is the SORT syntax needed for SPSS 12 without ADDVARIABLES subcommand.

```
SET MESSAGES=ON /WORKSPACE=768000.  
GET  
FILE='E:\XXX\SPSSextract.sav'.  
SORT CASES BY LEA Estab.  
AGGREGATE OUTFILE='E:\XXX\aggr.sav'  
/PRESORTED  
/BREAK=LEA Estab  
/DOB_mean = MEAN(DOB) /EntryDate_mean = MEAN(EntryDate).  
MATCH FILES  
/FILE=*  
/TABLE='E:\XXX\aggr.sav'  
/BY LEA Estab.  
EXECUTE.
```

* **Sort-13** – This is the SORT syntax needed for SPSS 13 and above that utilizes the ADDVARIABLES subcommand.

```
SET MESSAGES=ON /WORKSPACE=768000.  
GET  
FILE='E:\XXX\SPSSextract.sav'.  
AGGREGATE OUTFILE=* MODE=ADDVARIABLES  
/BREAK=LEA Estab  
/DOB_mean = MEAN(DOB) /EntryDate_mean = MEAN(EntryDate).
```

About SPSS Inc.

SPSS Inc. (NASDAQ: SPSS) is the world's leading provider of predictive analytics software and solutions. The company's predictive analytics technology improves business processes by giving organizations consistent control over decisions made every day. By incorporating predictive analytics into their daily operations, organizations become Predictive Enterprises—able to direct and automate decisions to meet business goals and achieve measurable competitive advantage.

More than 250,000 public sector, academic, and commercial customers rely on SPSS technology to help increase revenue, reduce costs, and detect and prevent fraud. Founded in 1968, SPSS is headquartered in Chicago, Illinois. For additional information, please visit www.spss.com.



To learn more, please visit www.spss.com. For SPSS office locations and telephone numbers, go to www.spss.com/worldwide.

SPSS is a registered trademark and the other SPSS products named are trademarks of SPSS Inc. All other names are trademarks of their respective owners.
© 2006 SPSS Inc. All rights reserved. SSRPWP-0506