Comparison of PROC SORTT, SyncSort, and PLSort

Janice M. B. Buck
Research Department
Pharmaceutical Division
CIBA-Geigy Corporation
Summit, New Jersey 07901

Section 1: Introduction

In recent years there has been discussion at SUGI on the difference between SyncSort* and PROC SORTT. In differing circumstances one sort was suppose to be significantly better than the other.

At our site there has been concern and effort to maximize the efficiency of our IBM 4361. Computer usage estimates showed that in the next year, usage could exceed the capacity of the 4361 and there were limited resources to upgrade. Much effort was put into streamlining the production programs for users. One area of concern were the number of sorts in some the statistical programs and whether these programs could be made more efficient by changing the sort.

In 1986, PLSort* was brought on site to test and compare. It was the first competitor we had to SyncSort and seemed to test well as a system sort. It was decided to compare PLSort's and SyncSort's performance inside SAS* programs. Since there had also been much interest in the efficiency of PROC SORTT for small data sets, all three sorts, PLSort, PROC SORTT, and SyncSort, were examined.

Section 2: The Methods

A comparison of four different sorts, PROC SORTT, SyncSort 6.0, SyncSort 5.3, and PLSort 86.1 was made. The parameters compared are the sort results, efficiency, and effect upon SAS programs. The indicators of efficiency are the CPU and memory used by SAS' Proc Sorts of data sets with varying numbers of observations. Four different jobs were run, one for each of the sorts, so that the results of each sort would be independent. Proc Options was run at the beginning of each program to show which sort was being used and to give a reference starting point for the status of the SAS program. The indicators of effect on the SAS environment are the CPU and memory used by SAS' Proc Prints of subsets of the sorted data sets. These results should show any degradation on the SAS environment if present. In addition, the four sorts were run with the two versions of the CMS SAS System, 82.3 and 5.16. The data sets used to check efficiency were sorted by a fixed numeric variable. This variable was randomly generated and was the only one on the data sets. The four different jobs were repeated upon different days and at different times to guarantee representative results.

Data sets containing both numeric and character data with the full assortment of characters and missing values were also sorted by the four sorts. The output data sets were compared for Proc Sorts without any options and for Proc Sorts with the following options: Reverse, Equals, Noequals, Noduplicates and the By statement option, Descending. Also, the output data sets from the efficiency and effect tests were compared.

Section 3: Results

The Table 1 shows typical results of the programs comparing the efficiency and effect of the four sorts, PROC SORTT, SyncSort 6.0, SyncSort 5.3 and PLSort 86.1.

The Proc Options at the beginning of each program shows that the original status of each job is similar and that a comparison of the results of the four sorts is valid. It shows as well that there is a difference in the efficiency in the 2 versions of the SAS System, with version 5.16 being more efficient, both CPU and memory wise.

The Proc Print's results do not significantly change for any job from the first Proc Print to the last. This implies that none of the sorts degrade the SAS environment with either version of SAS.

The results of the Proc Sorts do show differences in the efficiency of the four sorts. With both SAS versions, the PLSort consistently uses less CPU than either PROC SORTT or SyncSort 5.3. With version 5.16, SyncSort 6.0 is considerably more efficient CPU-wise than SyncSort 5.3, being about as efficient as PLSort. This difference is not seen with SAS version 82.3. SyncSort gains this CPU effect by using more memory in comparison with SyncSort 5.3. The four sorts use the same amount of CPU for 112 observations, but by 1000 observations there are significant differences between some of the sorts, with PROC SORTT being one of the least efficient. This is surprising as PROC SORTT is reportedly more efficient for small data sets up to about 2000 observations. Graphs 1 and 2 for SAS versions 5.16 and 82.3, respectively, demonstrate the above results.

All four sorts use considerably less memory with

Whereas, PLSort and SyncSort 5.3 use about the same amount of memory, SyncSort 6.0 uses more memory but sorts faster. One item of interest is that PLSort uses increasing amounts of memory as the number of observations increases, but the other three sorts use about the same amount of memory regardless of the number of observations. Graphs 3 and 4 illustrate these points.

For the data sets to compare efficiency and effect, the sorts yield data sets in the same order. The Table 2 shows the results for the data sets with numeric and character data containing all possible missing values. The Proc Sorts with no options and with the option, Descending on the By statement yield the same results. It is surprising to discover that many of the options for Proc Sort are unsupported by PROC SORTT. With version 82.3, Reverse, Equals, Noequals, and Noduplicates cause errors; with version 5.16, they cause warnings. For all Proc Sort options, PLSort and both versions of SyncSort produce the same output data sets.

Section 4: Conclusions

As a system sort, SyncSort 6.0 is about 50% faster than PLSort 86.1, which is significantly faster than SyncSort 5.3. SAS Proc Sort does not show this much difference, and actually yields the same efficiency for SyncSort 6.0 and PLSort 86.1 under SAS version 5.16. It would seem that if SAS effectively interfaces with each of the sorts this efficiency would occur in SAS programs as well.

None of the sorts seem to degrade the SAS environment. PLSort and SyncSort yield the same output data sets both for numeric and character data with all missing values. Looking at these facts, and the above mentioned efficiency of SyncSort 6.0 as a system sort, our site is continuing with SyncSort. We have not yet installed SyncSort in shared segments, which version 6.0 allows. We expect that, when we do, to see an even greater increase in SyncSort 6.0's efficiency.

The technical representatives at SAS Institute do warn that the efficiency of the PROC SORTT degrades with increasing number of variables on the data set. A further look at the four sorts would include records which are of longer length (about 300-1000 bytes) and with the records sorted based upon more than one field and fields of varying types.

REFERENCES

Squillace, Daniel J., Performance and Capacity Planning Considerations for the SAS System on Mainframes, Proceedings of the Eleventh Annual SAS Users Group International Conference, February 1986.

SAS Version 5.16

SORT	PROC OPTIONS	OBS=112 SORT	OBS=112 PRINT	OBS=1000 SORT	OBS=1000 PRINT	OBS=5000 SORT	OBS=5000 PRINT	OBS=10000 SORT	OBS=10000 PRINT	OBS=50000 SORT	OBS=50000 PRINT	OBS=100000 SORT	OBS=100000 PRINT
SAS SORT		0.23S	0.64S	0.48S	1.09S	0.49S	3.75S	0.50S	7.22S	0.50S	56.58	0.51S	114.57 .52S
		236K	1320K	236K	1320K	236K	1320K	236K	1320K	236K	1320K	236K	1320K 236K
PLSORT		0.19S	0.43S	0.52S	0.59S	0.52S	1.71S	0.52S	3.12S	0.53S	15.09	0.53S	30.13 0.53S
		236K	364K	236K	428K	236K	620K	236K	812K	236K	748K	236K	940K 236K
SYNC 6.0		0.23S	0.86S	0.49S	0.65S	0.50S	1.76S	0.51S	3.22S	0.52S	15.67	0.52S	31.37 0.52S
		236K	1324K	236K	1324K	236K	1324K	236K	1324K	236K	1324K	236K	1324K 236K
SYNC 5.3		0.22S	0.79S	0.54S	1.11S	0.55S	3.18S	0.55S	5.91S	0.53S	28.21	0.53S	57.27 0.52S
		236K	748K	236K	748K	236K	748K	236K	748K	236K	748K	236K	748K 236K

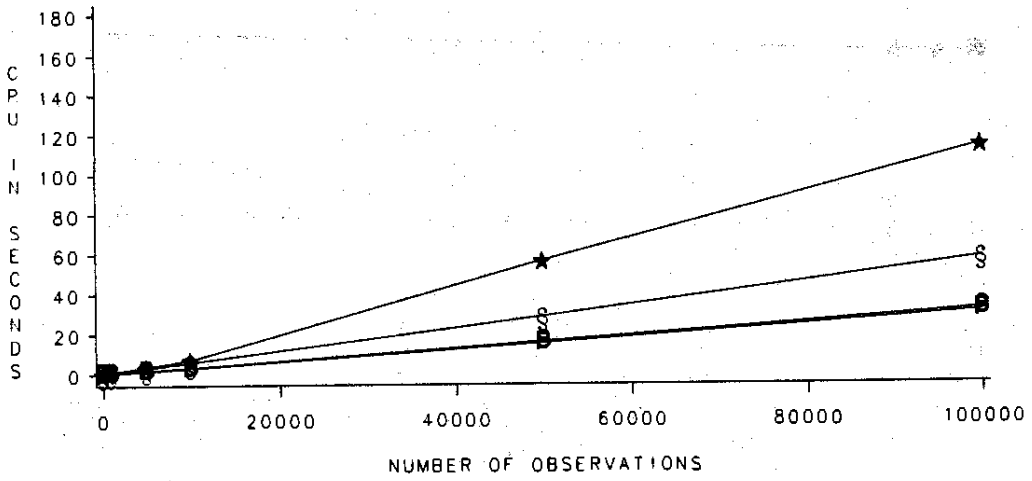
SAS Version 82.3

SORT	PROC OPTIONS	OBS=112 SORT	OBS=112 PRINT	OBS=1000 SORT	OBS=1000 PRINT	OBS=5000 SORT	OBS=5000 PRINT	OBS=10000 SORT	OBS=10000 PRINT	OBS=50000 SORT	OBS=50000 PRINT	OBS=100000 SORT	OBS=100000 PRINT
SAS SORT		0.52S	0.46S	0.47S	1.71S	0.44S	7.58S	0.45S	14.99	0.47S	86.54	0.46S	176.2 0.46S
		832K	2048K	832K	2048K	832K	2048K	832K	2048K	832K	2048K	896K	2048K 896K
PLSORT		0.49S	0.43S	0.47S	0.62S	0.48S	1.60S	0.48S	2.84S	0.49S	13.80	0.49S	28.03 0.48S
		832K	960K	832K	960K	832K	1152K	832K	1408K	832K	1344K	832K	1472K 832K
SYNC 6.0		0.51S	0.69S	0.47S	0.98S	0.47S	2.49S	0.47S	4.58S	0.47S	22.28	0.49S	46.22 0.48S
		832K	1792K	832K	1792K	832K	1856K	832K	1856K	832K	1856K	832K	1856K 896K
SYNC 5.3		0.50S	0.74S	0.47S	1.05S	0.47S	2.65S	0.47S	4.81S	0.48S	23.28	0.48S	47.27 0.48S
		832K	1344K	832K	1344K	832K	1344K	896K	1344K	896K	1344K	896K	1344K 896K

TABLE 1

Comparison of PROC SORTT, SyncSort, and PLSort

VERSION=5.16

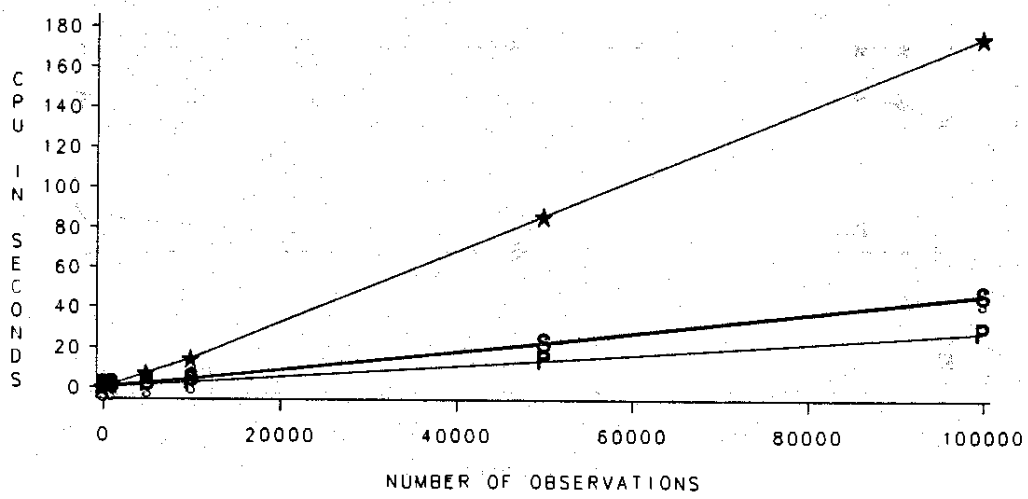


SORT P P P PLSORT 86.1 ★ ★ ★ SORTT
§ § § SYNCSORT 5.3 S S S SYNCSORT 6.0

Graph 1 By Janice M. B. Buck

Comparison of PROC SORTT, SyncSort, and PLSort

VERSION=82.3

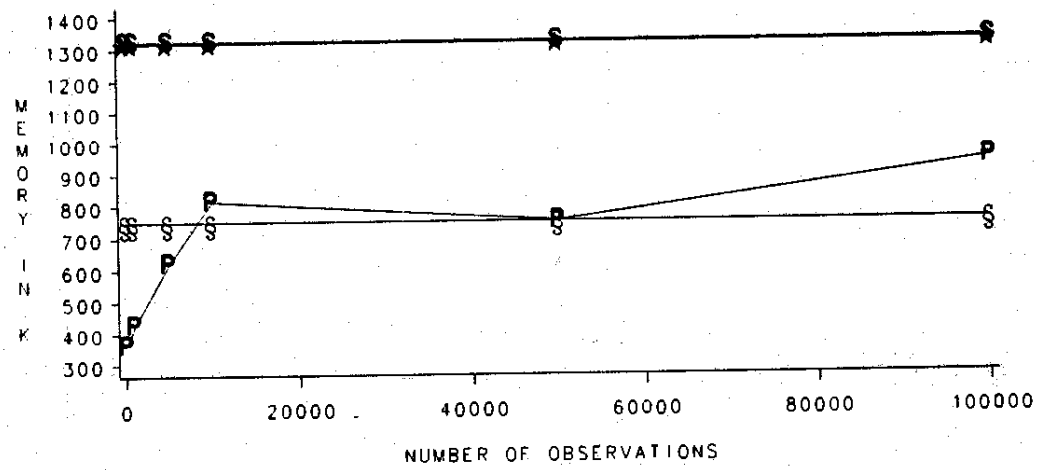


SORT P P P PLSORT 86.1 ★ ★ ★ SORTT
§ § § SYNCSORT 5.3 S S S SYNCSORT 6.0

Graph 2 By Janice M. B. Buck

Comparison of PROC SORTT, SyncSort, and PLSort

VERSION=5.16

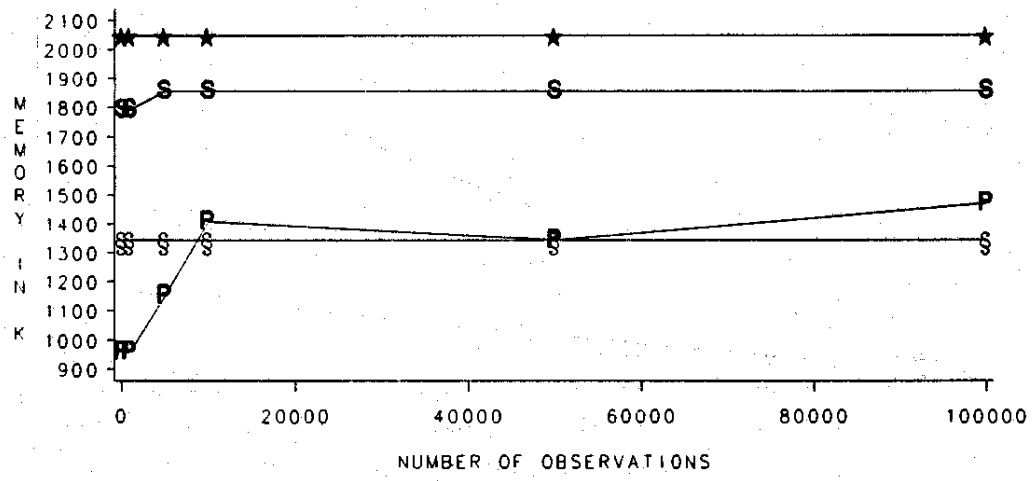


SORT P P P PLSORT 86.1 ★ ★ ★ SORTT
§ § § SYNCSORT 5.3 § § § SYNCSORT 6.0

Graph 3 By Janice M. B. Buck

Comparison of PROC SORTT, SyncSort, and PLSort

VERSION=82.3



SORT P P P PLSORT 86.1 ★ ★ ★ SORTT
§ § § SYNCSORT 5.3 § § § SYNCSORT 6.0

Graph 4 By Janice M. B. Buck

Order of Character and Numeric Data with Missing Values

Sort Option	SAS Version 5.16	SAS Version 82.3
None	same	same
Descending	same	same
Reverse	PLS=SYNC (not supported with PROC SORTT)	PLS=SYNC (error invalid option with PROC SORTT)
Equals	same (not supported with PROC SORTT)	same (error invalid option with PROC SORTT)
Noequals	PLS=SYNC (not supported with PROC SORTT)	PLS=SYNC (error invalid option with PROC SORTT)
Noduplicates	PLS=SYNC (not supported with PROC SORTT)	PLS=SYNC (error invalid option with PROC SORTT)

TABLE 2