Paper 096-2008

Poor man's Parallel Processing using the DATA Step View

Erik W. Tilanus, consultant, Driebergen, the Netherlands

ABSTRACT

The use of Data Step Views can reduce the elapsed time of programs that contain combinations of DATA steps and PROC steps. Under circumstances this reduction can be even in the order of 40 or more percent. The penalty is a higher CPU usage. The paper explains how Data Step Views are created and used and demonstrates the elapsed time saving, based on a practical example.

INTRODUCTION

In the SAS documentation the DATA Step View (DSV) is presented as just another view and compared to the data view that you can create e.g. with database management systems.
It is mentioned that the use of a DSV can reduce I/O activity and the need for workspace. Disadvantages are that the DSV is not platform independent and adds to processing overhead.
The DSV has however another advantage: it can reduce elapsed times of combinations of steps. I used the following program to prove the concept:

```
libname hv 'e:\sasdata';
data _null_;
t=time();
put t;
run;
data hisbooking;
set hv.hisbooking;
run;
proc sort data=hisbooking out=sortedbooking;
by dep arr class flight_date flight;
run;
data _null_;
t=time();
put t;
run;
```

The first and last DATA step are just for recording the elapsed time, the other DATA step and the PROC SORT is what it is all about. It just copies a data set to the work library and sorts it. My test data set happened to contain airline booking information, but the contents is of course irrelevant for the test.
I ran this program seven times, kicked out the slowest and fastest run (you never know what else is going on in a Windows environment that may slow down your program) and averaged the elapsed times and CPU times for the other five runs. On my computer system the resulting average elapsed time was 31.61 seconds and the average CPU time 3.05 seconds. Then I changed the DATA step into a Data Step View as follows:

```
data hisbooking/view=hisbooking;
set hv.hisbooking;
run;
```

With this DATA step I ran the program another seven times, kicked out the extremes and the result was an average elapsed time of 21.94 seconds, a reduction of 31%! The penalty is in the CPU time: the total recorded CPU time was on average 4.13 seconds, 35% higher than in the first test, but there might be some double counting involved.

SOURCE OF THE SPEED GAIN

Where does this speed gain come from? In fact there are two sources. Firstly, SAS, and especially the DATA step are very “I/O bound”, i.e. the elapsed time of a program depends much more on the I/O volume (and consequently HD speed and channel load) than it depends on CPU speed. By using DSV's you are eliminating the creation of a complete data set and therefore eliminating lots of I/O activity. Secondly, with the “classic” method, the data set has to be completed before the PROC SORT can start. Using the DSV, the input for the PROC SORT is created in parallel to the creation of the sort buffers, effectively creating parallel processing.

In the example at the end of the paper we will demonstrate a situation that goes even further: it runs two merges at the same time rather than sequentially!

CREATION OF THE DATA STEP VIEW

The creation of a DATA step View is simple: add the VIEW option to the DATA statement. The name in the VIEW option should be identical to the name of the data set you simulate in the view, including the data library, like in:

```
data hisbooking/view=hisbooking;
```

or

```
data hv.hisbooking_v/view=hv.hisbooking_v;
```

You cannot have a data set and a view of the same name in the same library. Neither can you create two views in one DATA step as you can create two (or more) data sets in one DATA step. If you specify two data sets in the DATA statement, like in:

```
data a b/view=a;
```

data set b will be created as a normal SAS data set, while executing the view.

So if you want both a and b to be a view, you should create two separate DSV's.

When you are creating DSV's just for the purpose of improving run times it is practical to store them in the work library. Then you don't have to worry about the standard message “NOTE: A stored DATA STEP view cannot run under a different operating system.” since it will be recreated every time you run the program. This is no problem since creating the view is lightning fast. On my system it normally takes in the order of 0.1 seconds!

WHEN TO USE

In principle it makes sense to use DSV's whenever you are processing data sets of considerable size and the output of the DATA step is only input for a next step and does not need to be stored. But a contra-indication is that you need the same data in more than one subsequent step, since then you are executing the view over and over again. So what is considerable size? That will vary depending on your system environment. My own rule of the thumb is any DATA step that runs more than some 10-20 seconds is a candidate if it meets the other requirements.

EXAMPLE

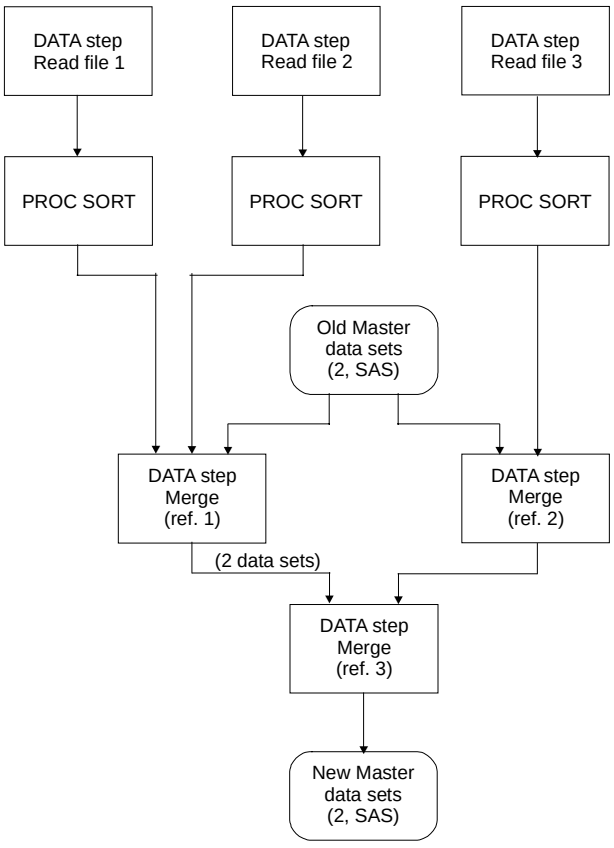
Recently I came across a situation where I had to read data from three different external files and combine the information with two existing master files, to create new master files.

Originally the program was set up using "ordinary" DATA steps. But as you can see from the diagram, you are creating six intermediate data sets in the various DATA steps, while you are only interested in the new Master data sets.
The runtime of this original program was 1 minute and 40 seconds, using 30.52 CPU seconds.

Next all DATA steps, except for the final merge to create the new master data sets, were replaced by DATA step Views. In DATA step (ref. 1) two output data sets are created, so it has to be duplicated: one view for each output data set.
This View-version of the program took only 51.51 seconds to complete, using 35.56 CPU seconds.

Table 1 presents timings per step, to show how the differences builds up. The DATA step views require only fractions of second to build. The figures in this table were taken from test-runs while no other applications were active.

As you can see the steps where the views are used consume more CPU time than their "classic" counterparts. In fact the CPU time does not only represent the time of the step itself, but also counts for the execution of the view.



In the final step (ref. 3) the elapsed time is higher than the final step in the version with normal DATA steps, which might seem be in contradiction of what has been stated before. However, running the same step again, while other applications were active and claiming their share of I/O activities the elapsed times changed: the "classic" step went up to 01:06.74 and the view-version went up to 45:54. This is a clear indication of the impact of the I/O overhead.

In summary, the results are impressive: elapsed time reduced by 49% (in the test with other applications active the gain was even higher: 65%!) with a penalty of 16% more CPU load.

Table 1: Timings per step

	Normal DATA Steps			DATA Step Views			Elapsed time difference build-up
	Real time	User CPU	System CPU	Real time	User CPU	System CPU	
Read File1	00:04.40	00:02.99	00:00.55	00:00.11	00:00.03	00:00.06	-00:04.29
Sort File1	00:03.38	00:00.83	00:00.48	00:05.27	00:03.69	00:00.55	-00:02.40
Read File2	00:01.98	00:00.70	00:00.14	00:00.05	00:00.03	00:00.03	-00:04.32
Sort File2	00:00.47	00:00.14	00:00.15	00:01.34	00:00.86	00:00.17	-00:03.45
Read File3	00:00.81	00:00.46	00:00.10	00:00.10	00:00.03	00:00.07	-00:04.16
Sort File3	00:00.21	00:00.06	00:00.06	00:00.75	00:00.52	00:00.10	-00:03.61
Merge (2 ds)	00:48.63	00:09.83	00:05.05	00:00.13	00:00.03	00:00.06	-00:52.11
				00:00.09	00:00.04	00:00.05	-00:52.02
Merge (1 ds)	00:02.55	00:00.47	00:00.13	00:00.09	00:00.03	00:00.05	-00:54.49
Final merge	00:37.45	00:05.19	00:03.20	00:43.59	00:24.26	00:04.89	-00:48.35
Total	**01:39.86**	**00:20.66**	**00:09.86**	**00:51.51**	**00:29.53**	**00:06.03**	**-00:48.35**

CONCLUSION

DATA Step Views are a simple and effective way to speed-up your SAS programs, especially in environments where the I/O transfer capacity is limited. Since the View is effectively executed in parallel with the step where the View is used, a kind of parallel processing is created, which may save considerably on elapsed time, although with a penalty of somewhat higher CPU loads. Try it – you'll like it!

REFERENCES

Further information about the DATA step View can be found in SAS 9.1 Language Reference: Concepts and SAS 9.1 Language Reference: Dictionary.

CONTACT INFORMATION

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

Name	Erik Tilanus
Address	Horstlaan 51
Postal code, City	3971 LB Driebergen
Country	the Netherlands
Work Phone:	+31 343 517007
Fax:	+31 343 517007
E-mail:	erik.tilanus@planet.nl
Web:	www.synchrona.nl

At the website you can also find other presentations by the author, held at previous SUGI and SAS Forum meetings.

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.