

Getting the Most From Your Host

David F. Harris, The American Cancer Society, Atlanta, GA

ABSTRACT

Remember the last time you ran out of disk space, memory, or time? As heavy-duty SAS® programmers, we always relish the opportunity to push our system to the max, and with few exceptions, the ®SAS system itself fosters this behavior/programming style. The insatiable appetite for resources leads to a rather unbecoming reputation among system capacity planners and administrators: *The SAS® system is a hog.*

However, this is only part of the picture. With a little effort, SAS® programmers can optimize not only their use of the system, but also the use of their own time. This paper explores some straightforward ideas, tips, and techniques for deciding how best to use your system without abusing yourself. Included in Part I of the presentation are a set of questions for determining when efficiency techniques are appropriate given the nature of the programming task, and a review of easy-to-use, often overlooked SAS® efficiencies that are readily available. In Part II,

we look at FSIZE, a SAS® program/CMS exec for determining disk space requirements.

Although this paper focuses on CMS systems, many of the tips are also applicable to MVS systems.

PART I - EFFICIENCY TIPS

QUESTIONS TO ASK BEFORE YOU CODE

It's always surprising to me how often SAS® programmers, including myself, are willing to subject ourselves to the brick wall of system

constraints. Like crash test dummies, no matter how fast we hit it, the wall always wins. However, unlike the dummies, SAS® programmers have options and option statements.

There are many tools that SAS® provides for avoiding the wall. But to bypass it, you first need to see it. And to see it, you need to know what signs you're looking for. Like getting a driver's license, there are some questions that need to be answered.

1. What are my limitations?
 - How much disk space is available?
 - How much CPU time is possible?
 - What are the memory constraints?
 - How many tape drives are available?
2. What are my own limitations?
 - How much do I know?
 - How much time do I have?
 - What resources do I have? (e.g., manuals, peers)
3. Do I even have a choice?
 - Is there
 - a constraint that requires efficient code?
 - an ethical, moral, or "managerial" obligation?
4. How often will the program be executed?
 - Is this a one time, ad hoc, report? Even so, if it's a system hogger, and if it might change a few times until the user is satisfied, then some efficiencies may be warranted.

5. Is a fast response a requirement?
If it's an interactive job, you ought to favor response time over disk space.
6. What benefits can be gained?
What costs can be avoided or reduced?
(Don't overlook 'individual' gains/costs.
If you take time to research an efficiency, you will likely be able to apply it again to another program.)
7. Where is the program's bottleneck(s)?
Not every line of code has to be optimized.
Spend time where it will have it the most effect.

TUNNELS, DETOURS, AND SHORTCUTS

When it comes to getting the most from your host, you need to realize that there's four main components in question: system time, system space, system memory, and, last but not less, *your* time.

When you consider overall efficiency, it's not unlike the naive card player, who, upon sitting down to the poker table, cautioned his fellow players that if they all played carefully, they could all win a little money.

Somewhere, somehow, the toll troll is going to get its due. For instance, if you're short on system memory, you may be able to break a job up into smaller components, but you're obviously going to take more time. That's why you need to keep the questions in mind. Of course, by the end of this paper, you should have a few answers in your mind, too.

Getting back to the brick wall, the object isn't to overpower it; it's to outsmart it. So here are some of my favorite tips, tricks, and techniques for scaling back your waits.

1. Consider compressing your data. Although the SAS® compression algorithm is not very sophisticated, you can reduce your space

requirements, especially if you can organize your data so that all numeric variables are hitched together, side by side. This is an output dataset option:
COMPRESS=YES

2. Use views. Way back in the olden days, any data that SAS® was going to read had to be translated into a SAS® dataset. If some other database created the data, you can now define a "VIEW" to that data, instead of reconstituting it as a SAS® dataset. This is a DATA step options:

DATA out.file / VIEW=out.view

3. Subset your data. Use **DROPS** or **KEEPS** to reduce the size of your data and this will save time.

4. Never, ever, run a yellow light. Any log that shows a data type conversion (i.e., character to numeric) should be evaluated and re-evaluated. This wastes system time, degrades your program's performance, and renders your data integrity extremely suspicious.

5. Use IF-THEN-ELSE rather than IF, IF, IF or IF ANDs. This will save processing time, especially if your organize conditional logic so that the most likely conditions are encountered early. Which leads to the next tip:

6. Know your data. Get a handle on the data with the many tools SAS® provides. A few **FREQs** and a **CONTENTS** will go a long way to helping you avoid unnecessary pitfalls.

7. Use the **IN** statement, rather than **ORs**, when possible. All **ORs** get rowed (i.e., are evaluated), but once you're in, you're in (i.e., the first true condition on an **IN** stops the evaluation process).

8. When updating a large master file with a transaction file, consider the merits of using

the **MODIFY** statement. This new feature edits the file in place, rather than creating a working copy. The statement is documented in *SAS® Technical Report P-242, SAS® Software: Changes and Enhancements, Release 6.08*.

9. Speaking of large master files—if the file is not often updated, it's a good candidate to be indexed. Indexing provides a means of accessing observations directly, rather than reading the dataset sequentially. This can save a good deal of time.

10. Know when, why, and how to use **WHERE**. Use them in PROCs that support them, especially PROC SORT. No need to sort the entire U.S. population when all your concerned with is Charleston's. Use them in DATA steps, the earlier, the better.

11. Check up on your system administrator. In CMS land make sure that SAS® is installed in saved segments. Check your CONFIG SAS file for a **SERIES=** statement. Also, make sure you're SORT options are aware of third party sort utilities that are installed at your site. You can check the *SAS® Companion for the CMS Environment* for details.

12. Compile your programs. After the bugs have been terminated and the illogic expunged, store your compiled program to save processing time. This is a DATA step option.

Create the stored program with

```
DATA SAS® / PGM=program-name;
    source program statements
RUN;
```

Access the stored program with

```
DATA PGM=stored-program-name;
REDIRECT INPUT|OUTPUT
    old-name-1=new-name-1 ...
;
```

RUN;

13. If your third-party sort support **NOEQUALS**, take advantage of it. The usual default for sort is to maintain the existing order. If this ordering is not needed, you can save a good deal of time in your jobs.

YOU'RE A RESOURCE, TOO!

Frequently, we hear ideas about saving system time, space, and memory, and forget about the most valuable resource of all: the programmer. So here are my top twelve tips for us.

1. Cheat! Use SAS/ASSIST® to generate quick examples of code which you can then save and manipulate. Similarly, use the SAS® on-line HELP. In fact, you can use the cut and paste to grab the format of an unfamiliar proc and drop it right into your program window.

2. If you're a SQL programmer, use **PROC SQL**. If you have a COBOL program to convert, get a copy of the SAS® routine from the Institute to semi-automatically convert the file definitions.

3. Learn how to use macros, and then don't! Actually, macros are most useful, highly helpful, and generally fun, once you get the hang of them. Just don't become so completely enamored that you end up encrypting your code without good reason. I once inherited a macro that went on for pages but I reduced it to 3/4 page of plain old DATA step code. On the other hand, I once saw a program that invoked the same 800 lines of input statements on three different files: 1598 lines of waste.

4. Here's an easy one: subscribe to SAS® *Communications* and SAS® *Observations*. Then take the time to read them when you get

them.

5. Invoke your on-line SUPPORT and peruse the changes and enhancements or use it to search for problems and solutions.

6. Install the SAMPLE libraries and get acquainted with the programs.

7. Know your PROCs. I've seen old programmers new to SAS® write a three page DATA step, and be proud that they could accomplish the task, having been used to pages and pages of coding in COBOL. But the three pages might have been reduced to three PROCs or maybe even one, if they took the time to peruse the manual.

8. Save yourself some heartache, headache, and eye strain by writing clear code, well indented and spaced, with good comments, and descriptive variable names.

9. Check your syntax first (e.g., OBS=0 or RUN CANCEL;). This will save you time and may save your data. Speaking of which, generate small, random sampled datasets to test your code against.

10. Network--Join your local SAS® users group. Monitor SAS® on the Internet. Go to SUGI. Attend SAS® class. Start or join an in-house users group. Share what you learn with your colleagues. Your employer should support all these activities as continuing education, especially if you come back with ideas that can help save your institution money and time.

11. Use your manuals. Buy a copy of SAS® *Programming Tips: A Guide to Efficient SAS® Processing*. Get on the Institute's Books-By-Users mailing list.

12. If you enjoy XEDIT more than DMS editor, do what I do: use it from within Display Manager. Here's a simple way to make it easy. First, set your PFKeys to CUA in you CONFIG SAS file. Second, set a DMS key to issue the X command (I use Shift-F3). In CMS, set a PFKey to issue the RETURN command (I use F3). For convenience, I set my DMS and CMS F12 keys to recall the last command SAS® syntax '?' and CMS 'RETRIEVE'). Lastly, I reduce my PROGRAM window to two lines and use the extra screen real estate for the LOG window. Now, by using F3 and SH-F3 I can easily move between DMS and CMS/XEDIT.

PART II – ESTIMATING DISK SPACE

Back at my office, disk space is always at a premium. Because we often deal with comparatively large datasets, it's important that programmers have a chance to estimate the size of their datasets. Because of this need, I have developed a small CMS EXEC and SAS® program. NOTE: Since we use FBA disks, the program would need to be modified for other disk architectures.

The program asks for the sum of the lengths of all the variables, the number of observations and the number of variable. It then presents a screen where responses can be modified (Example 1).

EXAMPLE 1

**** VERIFY FIELDS ****

```
BLKSZ = 1024 (Default=1K)
LRECL = 10 (Sum of lengths)
NOBS  = 10 (Number of obs)
NVAR  = 10 (Number of
           variables )
```

If you wish to change a field,
enter 'CHANGE ffff' where
ffff=field name.
Else press 'ENTER' if
you're happy.

The EXEC forwards this information into a SAS® program (Example 2), and the program is submitted. (The variable prefixed by '&' are passed from the EXEC into the SAS program.)

EXAMPLE 2

```
DATA MDSIZE;

SIZE=CEIL((98+372+5
+ (&NVAR * 82)
+ (&NOBS * (&LRECL
+ 4)))
/ &BLKSZ )
;

PROC PRINT;

RUN;
```

After the job runs, the output from the PROC PRINT is returned to the user in CMS BROWSE. The formula for this information is derived from the *SAS® Companion for the CMS Environment, Version 6, First Edition*.

CONCLUSION

In review, it's important to bear in mind that efficiency is almost always a relative ideal. Techniques for decreasing response time may lead to increased disk utilization; tips that help control disk space may use more system memory. The basic advice I can offer is take some time to get to know your tools, your system, and your data before you get going down the RUN way.

ACKNOWLEDGEMENTS

SAS® and SAS/ASSIST® are registered trademarks or trademarks of SAS® Institute Inc in the USA and other countries.

CMS is a registered trademark or trademark of International Business Machines Corporation.

REFERENCES

SAS® Institute Inc., *SAS® Companion for the CMS Environment, Version 6, First Edition*, Cary, NC: SAS® Institute., 1990. 319 pp.

SAS® Institute Inc., *SAS® Programming Tips: A Guide to Efficient SAS® Processing*, Cary, NC: SAS® Institute., 1990. 155 pp.

SAS® Institute Inc., *SAS® Technical Report P-242, SAS® Software Change and Enhancements, Release 6.08*, Cary, NC: SAS® Institute., 1993. 211 pp.