Paper 237-2012

The SAS® Log: A Wealth of Data and Job Flow Information

Steven First, Systems Seminar Consultants, Madison, WI USA

ABSTRACT

There have been many programs and papers written about reading the SAS® log for error checking and warnings. Although that information is valuable, we have found that the data messages that include filenames, row counts, and other information are a wonderful source of run-time metrics. This is where the real value of the SAS log is. These metrics, along with metrics provided by PROC SCAPROC, can be used to automate and validate run results, debug errors, provide job flow information and job optimization, and enable grids. This paper discusses the jobs that we have developed to use this neglected source of information.

INTRODUCTION

The SAS Log is a file that is produced by every SAS job, and it contains source listings, warnings, messages, errors, but also run times and row and variable counts. The SAS log should be viewed for any important SAS job to ensure that there were no errors or unexpected results in the current run. Several good papers have been presented about scanning the logs for errors and reporting them and even notifying users when jobs were in error.

Going through SAS source to find data flow, and using that source alone is very limited, as the SAS compiler makes many assumptions not evident from source. It occurred to me that the log was a richer source of information than the source alone.

To find errors, logic problems, slow running jobs, and verifying that results were correct, I found myself manually pouring through SAS test logs, quite often in parallel with a production log to dig out the record and variable counts to make sure that results were correct. A very smart person once told me, “We have computers, let’s use them!” Hopefully a tedious job of extracting row and column counts and run times from thousands of lines in a log can be automated. After the metrics are extracted, this process can feed an automated way to provide audit reports for run time checkout, to identify expensive steps, and also show the flow of data through the run.

What we often encounter with big systems and conversions, is that many tasks can be handled by computers, and reading the SAS logs is one of them that unlocks a rich database of run time activity.

INPUT TO THE SAS LOG READER

The SAS log file is main input file that is fed into our Log reader. In addition to the SAS log, other logs and files may be available such as Z/os Job logs and the Z/os JCL decks and the SAS source files themselves. All of these logs and inputs can be rerouted to a file in a variety of ways and then processed as a free form text file. I will show the data gleaned from JCL at the end, but the primary file again is the SAS log. The abbreviated log shown below is typical and not without its challenges to read. SAS of course is excellent at reading free form input, and by concentrating on the timings and row and column counts, the task is formidable, but doable in SAS. Some of the things that can make this process difficult are:

1. User options can turn off, minimize, or discard logging.
2. Different platforms gather statistics unique to that platform.
3. The system should be able to read logs from any SAS supported platform. Messages and notes do vary between the systems.
4. In a SAS/Connect or GRID environment, multiple logs are generated and often are on different platforms.
5. There is no guarantee that the log format won’t change in future releases.
6. Macro invocations can alter the layout of the log somewhat.

There are also some limitations to what is contained in the log.

1. While most steps report rows read in though PROC SQL does not.
2. Number of variables is reported on output files but not input files.

3. Files processed with user exits or SCL I/O may not report row counts.

Some (but not all) of these issues are addressed by PROC SCAPROC and will be mentioned later in this paper.

The bolded items in the SAS log are the ones that we are interested in.

```
NOTE: Copyright (c) 2002-2010 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.3 (TS1M0)
      Licensed to SYSTEMS SEMINAR CONSULTANTS INC, Site 70007546.
NOTE: This session is executing on the W32_7PRO  platform.

NOTE: SAS initialization used:
      real time           2.43 seconds
      cpu time            1.59 seconds

      . . .
46   data steps(keep= /*jcl_stepno jcl_stepname*/
47                    sas_step_no Actual_log_line sas_log_line        /*step_nm_no*/
48                    mprocname rsecs csecs u_csecs s_csecs excp)
49        datafile(keep=sas_step_no memtype i_o path engine ddn
                     member pathtype dobs dvars)
50        ddname_list
51        entire_log(keep=entire_line)
. . .     . . .
842       run;

NOTE: 78994 records were read from the infile "T:\clients\XYZ_Batch2_Log.txt".
      The minimum record length was 0.
      The maximum record length was 112.
NOTE: Mathematical operations could not be performed at the following places. The
results of the operations have been set to missing values.
      Each place is given by: (Number of times) at (Line):(Column).
      2 at 46:228   4 at 50:30
NOTE: The data set WORK.DDNAME_LIST has been updated.  There were 0 observations
rewritten, 5 observations added and 0 observations deleted.
NOTE: The data set WORK.STEPS has 3853 observations and 9 variables.
NOTE: The data set WORK.DATAFILE has 6672 observations and 10 variables.
NOTE: The data set WORK.ENTIRE_LOG has 78986 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time           0.81 seconds
      cpu time            0.39 seconds
```

PROCESSING THE SAS LOG

A series of SAS data steps read the free form SAS log using many INPUT features that SAS contains. The coding is somewhat difficult, as there are many nuances to the SAS logs, but in the end the results are quite good and dependable.

RESULTS OF THE LOG READER

There are two major types of files and corresponding reports that are produced. The first file contains step information from each unique SAS step the log lists, and this file can be used to show step level statistics, such as elapsed time, system time and so on. These runtime statistics can be sorted and ranked to identify the longest running steps for example.

The second group of reports is at a data file level, showing as many characteristics as possible for each file read or written. This file can be used to audit that correct row counts are read and written and also shows in a text file the flow of the information.

A LOG READER CANDIDATE

The report shown below was extracted from a SAS log containing almost 4000 steps and 80,000 lines in the SAS log. The client was having unusually long run times (one hour) with a job needing to be run hundreds of times. The run time needed to be reduced. The job needed to read billions of rows, and there is always a good chance that a few steps were causing most of the runtime problems. Complicating things was the fact that this job was GRID enabled and was actually being split up with pieces running on different servers, each with their own logs.

Error! Reference source not found. shows a few of the records of the Steps file.

Steps

Obs	sas_ step_no	Actual_ log_line	Sas_log_ line	mprocname	rsecs	csecs
1	0	6	.	PRINTTO	0:00:00.00	0:00:00.00
2	1	17	.	DATA	0:00:00.00	0:00:00.00
3	2	48	.	DATA	0:00:00.03	0:00:00.00
4	3	59	.	DATA	0:00:00.00	0:00:00.00
5	4	84	.	DATA	0:00:00.02	0:00:00.00
6	5	124	.	Initialization	0:00:00.05	0:00:00.03
7	6	145	.	PRINTTO	0:00:00.00	0:00:00.00
8	7	170	.	DATA	0:00:00.01	0:00:00.00
9	8	248	.	DATASETS	0:00:00.08	0:00:00.02
10	9	260	.	DATA	0:00:00.00	0:00:00.00
11	10	268	.	PRINTTO	0:00:00.00	0:00:00.00
12	11	285	.	DATA	0:00:00.00	0:00:00.00
13	12	300	.	DATA	0:00:00.00	0:00:00.01
14	13	315	.	DATA	0:00:00.03	0:00:00.03
15	14	355	.	DATA	0:00:00.02	0:00:00.01
16	15	374	.	CONTENTS	0:00:00.00	0:00:00.00
17	16	467	.	DATA	0:00:00.00	0:00:00.01
18	17	484	.	DATA	0:00:06.77	0:00:04.52
19	18	512	.	DATA	0:00:07.17	0:00:04.85
20	19	539	.	SUMMARY	0:00:03.05	0:00:02.45
21	20	565	.	SUMMARY	0:00:00.03	0:00:00.04
.	.	.	.	.	.	.
3853	3852	78972	.	DATA	0:00:00.06	0:00:00.00
					1:00:41.56	0:22:53.12

Output 1. The Steps data set

SUMMARIZING THE STEPS DATASET

It is easy to summarize the above file or rank the steps based on the run time and display the top 10 or so longer running steps. At first, the report was sorted with the longest running steps first, but then referring back to the original job was more difficult as the expensive steps were located all over the job. It was easier to just list the top 10 expensive jobs but list them in original order to expedite finding them. From the report, we see that almost 50% of the elapsed time is taken by the 10 most expensive steps and 21.47% taken by a single step. Clearly those are the steps to start with and in this case the expensive step was a PROC SUMMARY that was consuming huge amounts of memory. A simple change to not keep as many combinations of class variables solved the probably immediately. That change, along with some simple to the other nine expensive steps allowed for a total reduction run time of 40%.

Steps

Report of Elapsed Time
Top 10 Slowest Steps, step order

Obs	sas_ step_no	Actual_ log_line	Sas_log_ line	mprocname	Rank for Variable rsecs	Elapsed Time	Percentage of Total Elapsed Time
1	544	19436	.	DATA	4	0:01:52.54	3.0904
2	546	19474	.	DATA	8	0:01:20.18	2.2018
3	3440	70215	.	SQL	9	0:01:15.75	2.0802
4	3441	70232	.	SQL	10	0:01:01.04	1.6762
5	3503	71623	.	DATA	5	0:01:35.88	2.6329
6	3524	72180	.	SQL	7	0:01:23.96	2.3056
7	3525	72205	.	SQL	2	0:04:39.40	7.6725
8	3526	72233	.	SUMMARY	1	0:13:01.95	21.4729
9	3532	72560	.	DATA	6	0:01:34.69	2.6003
10	3540	72698	.	SORT	3	0:02:17.36	3.7720
						0:30:02.75	49.5049

Output 2. Report of Slowest Steps

Another way to show the difference is to draw a bar chart showing the same information in a graphical form.

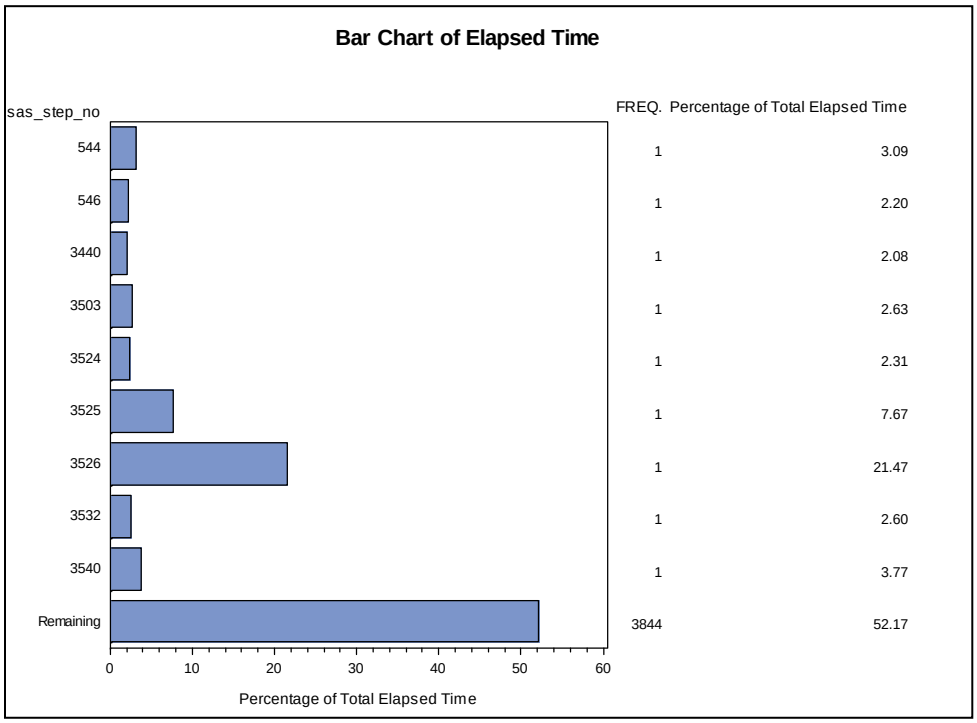

Figure 1. Bar Chart of Elapsed Time

THE JOB_FLOW DATASET

In many ways, the detailed Job_flow dataset is the most useful and exciting. Just as SAS programmers look through logs looking for row and variable counts and flow of the data, this dataset contains a row for virtually every file read or written into SAS. Because SAS doesn't report variables coming into a step, input datasets do not have the number of variables. This, along with the names and other attributes of columns, can be captured from PROC SCAPROC output and interleaved. For now, we still have a very useful report showing much information from the over 8000 files accessed in this system. We have developed run time summary reports reporting all of the relevant statistics from production runs for example for an easy audit of how things ran. There are many more ideas that we have to exploit this data, such as automatically generating data flow diagrams from this file.

job_flow

step	line	mprocname	engine	ddn	member	memtype	i_o	dobs	dvars
14	355	DATA	V9	SASDATA	VMACRO09	SAS DS	In	279	.
15	374	CONTENTS		WORK	FIELDS	SAS DS	Out	23	2
16	467	DATA		WORK	FIELDS	SAS DS	In	23	.
17	484	DATA	V9	SASDATA	C10TEST_WI	SAS DS	In	4,457,841	.
	484	DATA	V9	SASDATA	C10TEST_WI	SAS DS	Out	4,457,841	26
.	.	.	.	.	.	.	.	.	.
3852	78972	DATA	V9	SASDATA	GSOFF	SAS DS	In	1	.
	78972	DATA	V9	SASDATA	GSOFF	SAS DS	Out	1	1

Output 3. The Job_flow data set

EXPANDING TO READ MANY LOGS

Another huge bookkeeping task was solved by our log reading program. We were developing a very large SAS system with over 100 programs. Each of the programs needed to be tested and validated for each of three runs. The first run was 1% sample of the data, the second a 10% sample, and finally a full 100% sample run was executed. Each of these jobs was run many times, while testing with each run generating a SAS log. While the programmer was responsible for eliminating errors, the log checking program pulled out observation counts that needed to match a production control dataset. That was fine for a single log, but there were thousands to wade through and only the most recent for each program was valid. It was my job to report progress of the entire project and final run times of each of the jobs.

By wrapping a macro around the log reader program, we were able to read the directory where the logs were stored, locate only the most recent log for each program/sample combination, and then run the log reader for all the appropriate logs. The results were appended into a final report, which filled in the run time, as each program was completed. Those not yet complete showed up as missing values and eventually the table was filled in. There is no way that this number of validations could happen manually at a reasonable cost without a program.

Output 4 below displays a subset of the final table.

Latest All Prod runs by Sample Id detail

	sample		
	1	10	100
	Elapsed Time	Elapsed Time	Elapsed Time
Program			
pgm100_create_sas_views	0:00:00	0:00:00	0:00:00
pgm100_xyxpull	0:00:03	0:04:22	5:39:20
pgm112_xyzpull_d_abc_part2	1:59:36	.	.
pgm112_xyzpull_d_abc_part3	10:39:23	0:00:19	0:20:38
pgm113_xyzpull_d_abc_part4	1:59:34	.	.
pgm113_xyzpull_d_abc_part5	8:14:31	0:00:19	0:20:54
.	.	.	.
pgm141_xyzpull_hview_part6	1:08:01	0:00:44	0:32:28
pgm991_final	.	0:00:01	0:11:18

Output 4. The System Summary Report

PROC SCAPROC

PROC SCAPROC was introduced a few years ago to extract some of the same information, plus more than the log reading program also extracts. The procedure had some problems extracting correct data, especially with jobs containing macro code. SAS release 9.3 has fixed most of the bugs, and it is much improved.

The source job must be altered to start PROC SCAPROC at the beginning of the job and close it at the end of each job. It works by inserting comments into the SAS log, which can then be extracted as our Log reader does. An example of a portion of a log using PROC SCAPROC is show in output 5.

```
/* JOBSPLIT: DATASET INPUT SEQ WORK.A.DATA */
/* JOBSPLIT: LIBNAME WORK ENGINE V9 PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS
Temporary Files\_TD1252 */
/* JOBSPLIT: FILE OUTPUT C:\winnt\profiles\userid\record.txt */
/* JOBSPLIT: SYMBOL GET SYSSUMTRACE */
/* JOBSPLIT: ELAPSED 2750  */
/* JOBSPLIT: PROCNAME MEANS */
/* JOBSPLIT: STEP SOURCE FOLLOWS */
proc means data=a;
run;
```

Output 5. A PROC SCAPROC Altered Log

Advantages to using PROC SCAPROC are:

1. It is maintained by SAS.
2. It has access to the SAS compiler, to detail SAS column information, and can get at run time information that we can't.
3. It inserts comments in the log that are easily identifiable.
4. If certain SAS products are installed, it can split a job into a grid ready job.

Disadvantages to using PROC SCAPROC are:

1. It requires altering your job and rerunning.
2. It does not report record counts.
3. Reading the comments still involves parsing text.

A good system would be one that uses PROC SCAPROC for what it is good for and other reading techniques for features not in the proc.

READING Z/OS JCL AND JOB LOGS

For mainframe users, JCL decks and job logs are another source of system run time features. SAS is excellent at reading free form input such as JCL which generally doesn't require operands to be in fixed columns, but rather just separated with spaces.

We have developed JCL reading modules to extract useful information for estimating conversions, cross referencing programs to data, and adding information to the other log reading output. Output 6 shows a sample of the data extracted from JCL.

jcl

Obs	Job	step	sname	Jcl_proc	jcl_pgm	Ddn	path	Disp	Lrecl	Recfm
1	JOB1	1	STEP005		IDCAMS	SYSPRINT	SYSOUT=*			
2	JOB1	1	STEP005		IDCAMS	SYSIN	*			
3	JOB1	2	STEP010	SASCURR		OUTSAS	XYZZ.X01.ABC.JOB1.STEP010.AA	(,CATLG)		
4	JOB1	2	STEP010	SASCURR		SYSIN	*			
5	JOB1	20	STEP100	SASCURR		OUTFILE	XYZ.X01.ABC.JOB1.EEEE	(,CATLG)	132	FB
6	JOB1	20	STEP100		ASCURR	SYSIN	*			
7	JOB1	21	STEP110		FTP	SYSTCPD	ABC.SYSLIB(MVS1TCP)	SHR		
8	JOB1	21	STEP110		FTP	DD1	DUMMY	SHR		

Output 5. JCL Extracted data

CONCLUSION

Some things we would like to do to the log reader program are:

1. Merge all data from JCL, system logs, SAS logs, and PROC SCAPROC into a single program.
2. Add a graphical data flow diagram as output.
3. Try and use the data for job optimization to eliminate redundant steps, unneeded steps and variables, suggest shorter column lengths.

RECOMMENDED READING

- http://www.sys-seminar.com/newsletter

CONTACT INFORMATION

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

Steven First
Systems Seminar Consultants
2997 Yarmouth Greenway Drive
Madison, WI 53711
608 278-0065
sfirst@sys-seminar.com
www.sys-seminar.com