MWSUG 2016 - Paper BB18

SAS Advanced Programming with Efficiency in Mind: A Real Case Study

Lingqun Liu, University of Michigan, Ann Arbor, MI

ABSTRACT

This paper uses a real work example to demonstrate the concept and some basic tips of SAS programming efficiency. The first section of the paper introduces the background of a SAS application and its performance metrics. The second section analyzes the structure and features of the SAS application. The third section analyzes the log of the application to identify efficiency issues. In addition, in this section a log analysis utility is introduced. The fourth section provides a re-developed version of the application with performance improved to reduce 99.6% of its runtime. The last section tries to raise awareness of SAS programming efficiency and suggests some basic tips. The application discussed in the paper has been tested with SAS 9.2, 9.3 and 9.4 on Windows machines. The target audience includes SAS programmers from beginner to advanced level.

INTRODUCTION

Did you ever have any SAS applications that took longer than you expected to run? It could be hours, or even days long. Most of us find it frustrating when things like that happened, especially when you had a tight deadline to meet, or you had to run the job many times within a limit of time. Many programmers might think it is caused by the nature of their SAS application, such as big data sets, complex process, and limitation of computing power and resources, etc. Moreover, it was not uncommon that many SAS application developers/programmers did not realize that there usually were efficiency issues. This paper uses an example to raise the awareness of SAS programming efficiency, introduce a log analysis utility, and provide some basic tips.

I.1 BACKGROUND

UM-KECC is a multidisciplinary research center within the UM School of Public Health (SPH). UM-KECC was formed in 1993 and its mission is "to promote health, improve clinical practice and patient outcomes, optimize resource utilization, and inform public policy regarding organ failure and organ transplantation." UM-KECC pursues this mission "through high quality research, advances in biostatistics, and post-graduate education and training." (www.kecc.sph.umich.edu).

UM-KECC has been working with CMS to develop quality measures of ESRD patient care for years. Each quarter, as one tiny part of the large efforts, UM-KECC produces lists of ESRD patients included in the dialysis facility compare (QDFC) measures for more than 6,000 Medicare dialysis facilities nationwide. There are five measures: M1, M2, M3, M4, and M5. In each quarter, there are more than 21.7K patient list files (21,870 for 201607, 21,702 for 201604) created. This whole process consists of five similar SAS jobs, one for each measure.

M1_DFC_Patient_Lists.sas
M2_DFC_Patient_Lists.sas
M3_DFC_Patient_Lists.sas
M4_DFC_Patient_Lists.sas
M5_DFC_Patient_Lists.sas

I.2 PROCESS TIME

The process time varies for the jobs. The M5 job took about 10 hours. (And it could occasionally even take longer than 69 hours for some reason in reality. It was the worst case we had!) The rest took from 18 seconds to around 16 hours. The total process times for the last two quarters were about 19.4 hours and 33.4 hours.

Jobs	201604		201607	
	Real time	CPU time	Real time	CPU time
M1_DFC_Patient_Lists.sas	4:16:02.04	4:04:10.54	4:11:53.18	4:04:35.88
M2_DFC_Patient_Lists.sas	16:30.83	13:40.53	1:48:09.24	15:27.02
M3_DFC_Patient_Lists.sas	1:39.45	18.93	2:30.09	22.88
M4_DFC_Patient_Lists.sas	4:49:30.74	4:42:13.26	16:17:07.14	7:34:12.81
M5_DFC_Patient_Lists.sas	10:02:30.96	9:49:39.17	11:02:17.87	10:22:32.44
total	**19:26:14.02**	17:08:55.50	**33:21:57.52**	22:17:11.03

II. CODE ANALYSIS

You may wonder why some of these simple jobs can take more than 10 hours. Moreover, 19 to 33 hours of total runtime of the production is way too long. Are there any efficiency issues? Can the application be improved? Let us start with examining the SAS code, in order to see what the issues could be and identify how to fix them. In the following sections, our analysis and redevelopment will use M5 job as an example. The rest of these jobs are identical in terms of the code design, structure, functionality, and issues, etc. Please see Figure 1.1 and Figure 1.2 for the code listings.

Figure 1.1 Original Code Snapshot One

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_original_MWSUG.sas *

```

proc sort data=faclib.facinfo_&lookupdt. out= facinfo (keep= facid network provname provcity state)
   where DFC_report=1;
   by facid;
run;

%macro print_list(data,measure,name);

%IF &measure=M5  %then %do;
%let vars=firsts dialysis_90days age_ge_18 calcium_uncorrected in_facility modality elig_pm avg_3mo
%let varslabel= firsts='First^service^date' dialysis_90_days='Dialysis^ge^90^days' age_ge_18='Patie
in_facility="Meets^facility^requirement" modality="Meets^modality^requirement" elig_pm="Eligible^pa
hypercal_gt10_2="Hypercalemia-gt^10.2";
%end;

%put &vars;
%put &varslabel;

*************************************************************************************;
*--------Sort measure files, keep only variables output to list--------------------------*;
*************************************************************************************;

proc sort data=mlib.&data. out=temp(keep=patid facid &vars year month quarter);
   by patid;
run;

/**********************************************************************************
*--Merge individual measure files with patients to get patient identifiers---*;
/*********************************************************************************/
data saflib.M5_plist_&dateit.;
   merge temp (in=a)  saflib.patients (keep=patid surname first_name m_initial ssn);
   by patid;
   if a;
   fname=trim(first_name)||' '||trim(m_initial);
   Patient_id=_n_;
   ssn1=ssn+0;
    **** Note: In this step, a small percentage of pts have characters in their SSN.  This    ****;
    *    causes warning messages in the log file because ssn1 cannot be calculated, and in the   *;
    *    final patient list they will have a missing SSN.  Since the SSNs are not numeric, we    *;
    *    assume they are not valid, so having missing SSN is not a problem.                      *;
run;

/**********************************************************************************
 *---Merge with facinfo to obtain provider name, city, state, etc---*
**********************************************************************************/
proc sort data=saflib.M5_plist_&dateit.;
   by facid;
run;

```

CODE LOGIC

It is a simple job and it has two requirements:

1. ***Create data***: Put facility information (6,499 observations), patient information (2,819,069 observation) and measure results (6,423,888 observations) together to create a patient-measure level data set containing information for patients included in the measure for all facilities. Also, perform a few data manipulations.
2. ***Print data:*** Print patient-measure information by facility in plain text format with file extension .txt.

Figure 1.2 Original Code Snapshot Two

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_original_MWSUG.sas *

```
data  &measure._ptlist;
   merge saflib.M5_plist_&dateit.(in=a)  facinfo (in=infacinfo);
   by facid;
   if a and infacinfo;
   facility=trim(provname)||', '||trim(provcity)||', '||state;
   format ssn1 ssn11.;
   report_period=strip(year)||"    "||strip(month)||"    "||strip(quarter);
run;


proc sort data= &measure._ptlist;
   by network facid facility surname first_name ;
run;


proc sql;
  select count(distinct facid) into: numprovs
  from  &measure._ptlist;
quit;

%put &numprovs;

data _null_;
   length numprovsc $9.;
   numprovsc=strip(&numprovs);
   call symput('numprovsc', numprovsc);
run;


proc sql;
  select distinct facid into :prov1  -:prov&numprovsc notrim
  from  &measure._ptlist;
quit;


%do i=1 %to &numprovs;
     %put '******************' &&prov&i;
      data prvlevel ;
         set  &measure._ptlist;
         where facid="&&prov&i";
         call symput ("facility", compress(facility,"'"));

      run;
      %put '******************' &facility;

     ods listing file="&outfile";
           title "CONFIDENTIAL: Patients included in the &name. measure reported in the";
                  title2       "Quarterly Dialysis Compare-Preview for &month., &year. report.";
                  title3 "MMM Certification Number=&&prov&i Facility=&facility";
     options  ls=max ps=85;
           proc print data=prvlevel noobs  split='^' uniform;
```

CODE DESIGN AND STRUCTURE

This code has two parts, one for each subtask. The first part consists of PROCs and DATA steps. The second subtask is implemented with a %MACRO %do loop that creates and prints out one data set for each facility. As a result, there are more than 6,000 DATA steps and PROCs generated by the %MACRO/%DO loop at runtime.

1. ***Create data:*** Four PROC SORTs, two DATA MERGEs.
2. ***Print data:*** Two PROC SQLs, one DATA _NULL_, one %MACRO %do loop of 1 DATA step and ODS/PROC PRINT.

SAS FEATURES

There are many SAS features, including some advanced ones, in this SAS application.

- DATA STEP MERGE, PROC SQL, PROC SORT;
- %MACRO, &&VAR&N, CALL SYMPUT, INTO:, %Do loop; DATA _NULL_;
- Data type conversion (+0), function COMPRESS(), STRIP(), TRIM();
- ODS LISTING, Dynamic titles, PROC PRINT options, etc.
- System options: LS, NODATE, NONUMBER, NOCENTER, ERRORS, SOURCE2, MPRINT.

CRITICAL THINKING

Does it need to be so complicated (using so many steps and features)? Is %macro really needed? (Can the %macro be avoided?) Which features/steps did take most of the runtime? Would the large number of small DATA steps and PROCs be an efficiency issue? Or is the long runtime due to the large size of the input SAS data sets? To answer these questions, I inspected the log files of the job along with the SAS code.

III.1 LOG ANALYSIS: OBSERVATION & ESTIMATION

The log file is lengthy. It has more than 45,000 lines. We need to search for the key words 'real time' to see how long each step took. First, let us look at the runtime for task one -- the creation of measure-patient data set. The facility info data has about 6,600 records. The measure data has about 6.5 million observations. The patient info data set has about 2.5 million records. The DATA step and PROC SORT processed these data sets within a few minutes. It is fast to create the measure-patient data set. Since SAS is so powerful, the sizes of the data sets in this application are not the issue (Please see Figure 2.1 and Figure 2.2 for details.)

Figure 2.1 Log Snapshot One

```
I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG (2).log
NOTE: There were 2819069 observations read from the data set SAFKECC.PATIENTS.
NOTE: The data set SAFLIB.M5_PLIST_201604 has 6423888 observations and 21 variables.
NOTE: Compressing data set SAFLIB.M5_PLIST_201604 decreased size by 42.01 percent.
      Compressed is 64233 pages; un-compressed would require 110757 pages.
NOTE: DATA statement used (Total process time):
      real time           1:33.78
      cpu time            37.01 seconds


SYMBOLGEN:  Macro variable DATEIT resolves to 201604
MPRINT(PRINT_LIST):   proc sort data=SAFLIB.M5_plist_201604;
MPRINT(PRINT_LIST):   by provfs;
MPRINT(PRINT_LIST):   run;

NOTE: There were 6423888 observations read from the data set SAFLIB.M5_PLIST_201604.
NOTE: The data set SAFLIB.M5_PLIST_201604 has 6423888 observations and 21 variables.
NOTE: Compressing data set SAFLIB.M5_PLIST_201604 decreased size by 42.00 percent.
      Compressed is 64234 pages; un-compressed would require 110757 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time           1:36.95
      cpu time            36.65 seconds

```

Figure 2.2 Log Snapshot Two

```
I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG (2).log *
NOTE: The data set WORK.M5_PTLIST has 6423888 observations and 27 variables.
NOTE: Compressing data set WORK.M5_PTLIST decreased size by 51.77 percent.
      Compressed is 88522 pages; un-compressed would require 183540 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time           2:47.55
      cpu time            1:04.28
```

Now let us look at the runtime for task two – the creation of the facility specific patient list files. Every time a list file was created, one small DATA step and one PROC PRINT were executed. After scanning the log file, we noticed that the process only used about 5.3 seconds or so per facility.

However, since there were more than 6,000 facilities, the total runtime ended up as about 10 hours. The stop value of the %DO loop was 6,375 for this case. Therefore, the total run time was about 5.28*6375/(60*60) seconds = 9.35 hours. (Please see Figure 2.3 and Figure 2.4 for details)

Figure 2.3 Log Snapshot Three

```
I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG (2).log
578  NOTE: There were 1404 observations read from the data set WORK.M[redacted]_PTLIST.
579        WHERE [redacted]';
580  NOTE: The data set WORK.PRVLEVEL has 1404 observations and 27 variables.
581  NOTE: Compressing data set WORK.PRVLEVEL decreased size by 53.66 percent.
582        Compressed is 19 pages; un-compressed would require 41 pages.
583  NOTE: DATA statement used (Total process time):
584        real time           5.28 seconds
585        cpu time            5.28 seconds
```

Figure 2.4 Log Snapshot Four

```
I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG (2).log
457571 MPRINT(PRINT_LIST):   ods listing close;
457572 MLOGIC(PRINT_LIST):  %DO loop index variable I is now 6376; loop will not iterate again.
457573 MLOGIC(PRINT_LIST):  Ending execution.
457574 294
457575 295
457576
457577 NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
457578 NOTE: The SAS System used:
457579       real time           10:02:30.96
457580       cpu time            9:49:39.17
457581
```

III.2 LOG ANALYSIS: STATISTICS

To get the statistics of the runtime of the SAS application, I developed a simple SAS utility to analyze the full lengthy SAS log file (457,581 lines in this case). The log analysis utility consists of two small %macros: %log_io_search(), %log_io_data() and a PROC MEANS. (Please see Figure 3.1 for details.)

Figure 3.1 Log Analysis code

```
Programmer's File Editor - [log_analysis_MWSUG.sas]
File  Edit  Options  Template  Execute  Macro  Window  Help

*************************************************************************;
*Program name: log_analysis.sas
*By          : lqliu@umich.edu 2008, 2016
*Purpose     : to process SAS log file to analyze SAS application
*            :   --structure and performance
*            :
*Input       : SAS log file
*Output      : two txt file and two datasets, plus ...
*
*Note        : internal use only
*            : some lines in step 3 need revision to reflect individual needs
****************************************************************************;
option mprint;
***************** STEP 1 ************************************************;
** usage: --------------------------------------------------------------;
** %log_IO_search(log= [your log file].log,
                  doc=[results txt file].txt);
************************************************************************;
%macro log_IO_search(log=,doc=);
%*if not %index(&log,'.') %then %let log=&log.*.log;
data _null_;
length logname f logline $200 ;
infile "&log" filename=f  end=done;
file "&doc";
logname=f;
if logname ne lag(logname) then do;
    if line then put line "lines read";
    put // '-----' logname '------';
        line=0;
end;
input ;
line + 1;
output = index(_infile_,'NOTE: The data set')  and
         not index(_infile_,'-- NOTE:')
                   or
         index(_infile_,'were written to the file')
;
input = index(_infile_,'read from')  or   index(_infile_,'WHERE ')  ;
time  = index(_infile_,' time')  ;
logline = _infile_;
keep= ifc(input,'INPUT ','OUTPUT') ;
keep= ifc(input,keep,'TIME  ') ;
if input or output or time then put keep logline;
if done then put line "lines read";
run;
%mend;

%let log=I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.log;
%let doc=I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.txt;
%log_IO_search(log=&log,doc=&doc);
```

```
;
input = index(_infile_,'read from')  or   index(_infile_,'WHERE ')  ;
time  = index(_infile_,' time')  ;
logline = _infile_;
keep= ifc(input,'INPUT ','OUTPUT') ;
keep= ifc(input,keep,'TIME  ') ;
if input or output or time then put keep logline;
if done then put line "lines read";
run;
%mend;

%let log=I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.log;
%let doc=I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.txt;
%log_IO_search(log=&log,doc=&doc);

******** STEP 2****************************************;
** usage: ----------------------------------------------------------;
** %log_IO_data(log= [results txt file from step 1 above].txt,
                   doc=[results txt file].txt);
**********************************************************************;
%macro log_IO_data(log=,doc=);
data log_runtime_messy log_runtime(keep= dsn ntime ctime procdat  obs);
length logname f logline $200 dsn $32 PROCDAT $6;
retain dsn obs;
infile "&log" filename=f  end=done;
 file "&doc";  *optional;
logname=f;
if logname ne lag(logname) then do;
    if line then put line "lines read";
    put // '-----' logname '------';
        line=0;
end;
input @;
 if
 index(_infile_,'TIME NOTE: The data set')
 or index(_infile_,'TIME NOTE: DATA statement used (Total process time):')
 or index(_infile_,'TIME NOTE: PROCEDURE SORT used (Total process time):')
 or index(_infile_,'TIME NOTE: PROCEDURE SQL used (Total process time):')

then
```

:\MWSUG16__BB18\log_analysis_MWSUG.sas

```
      input /;
      ctime = scan(_infile_,4,'');
      if index(ctime,':') then do;
         if countc(ctime,':')=1 then ctime='0:'||ctime ;
         ntime=input(strip(ctime), time11.2);
      end;
      else ntime=ctime+0;
          if PROCDAT='DATA: ' then do; DATA_TIME+ntime; DATA_steps+1; end;
          else if PROCDAT='SORT: ' then do;SORT_TIME+ntime; SORT_steps+1; end;
          else do;SQL_TIME+ntime; SQL_steps+1;end;
      output;
       put PROCDAT _infile_  @46 OBS comma10.0 ' ' @60 DSN  "---" ntime=mmss8.2;  *optional;
     end;
   else input;
 end;
 else input;
 if done then do;
      put DATA_steps " DATA steps -- total process time " DATA_TIME=time11.2  ;  *optional;
          put SORT_steps " SORT steps -- total process time " SORT_TIME=time11.2; *optional;
          put SQL_steps " SQL steps -- total process time " SQL_TIME=time11.2  ;   *optional
end;

run;
%mend;
option mprint;
 %let log=I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.txt;
 %let doc=I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG2.txt;
 %log_IO_data(log=&log,doc=&doc);

** STEP 3 ***********************************************************;
** summarize the results;
*********************************************************************;
proc means data=log_runtime   mean max min sum;
class procdat dsn ;
var ntime obs;
types dsn procdat procdat*dsn ;
run;

** ENDSAS ***;
```

The first macro %log_io_search() uses a DATA _NULL_ step to search through the log file, extract the key information for each step, and write out them into a txt file. (Figure 3.2)

Figure 3.2 Log Analysis Results Snapshot One

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.txt

```
-----I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.log ------
TIME real time              0.10 seconds
TIME cpu time               0.06 seconds
INPUT NOTE: There were 6553 observations read from the data set FACLIB.FACINFO_201601.
INPUT WHERE DFC_report=1;
TIME NOTE: The data set WORK.FACINFO has 6553 observations and 5 variables.
TIME NOTE: PROCEDURE SORT used (Total process time):
TIME real time              0.74 seconds
TIME cpu time               0.06 seconds
INPUT NOTE: There were 6554484 observations read from the data set MLIB.M5_PATIENT_LIST.
TIME NOTE: The data set WORK.TEMP has 6554484 observations and 14 variables.
TIME NOTE: PROCEDURE SORT used (Total process time):
TIME real time              19.53 seconds
TIME cpu time               18.54 seconds
```

The second macro %log_io_data() again uses a DATA step to search through the output text file generated from the first step and put the results in a better text format. (Figure 3.3)

Figure 3.3 Log Analysis Results Snapshot Two

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG2.txt

```
-----I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.txt ------
SORT: TIME real time           0.74 seconds          6,553    WORK.FACINFO ---ntime=0:00.74
SORT: TIME real time           19.53 seconds     6,554,484    WORK.TEMP ---ntime=0:19.53
DATA: TIME real time           27.00 seconds     6,554,484    SAFLIB.M5_PLIST_201607 ---ntime=0:27.00
SORT: TIME real time           18.59 seconds     6,554,484    SAFLIB.M5_PLIST_201607 ---ntime=0:18.59
DATA: TIME real time           26.34 seconds     6,554,484    WORK.M5_PTLIST ---ntime=0:26.34
SORT: TIME real time           29.15 seconds     6,554,484    WORK.M5_PTLIST ---ntime=0:29.15
SQL : TIME real time           7.20 seconds      6,554,484    WORK.M5_PTLIST ---ntime=0:07.20
DATA: TIME real time           0.00 seconds      6,554,484    WORK.M5_PTLIST ---ntime=0:00.00
SQL : TIME real time           6.75 seconds      6,554,484    WORK.M5_PTLIST ---ntime=0:06.75
DATA: TIME real time           5.21 seconds          1,380    WORK.PRVLEVEL ---ntime=0:05.21
PRNT: TIME real time           0.01 seconds          1,380    WORK.PRVLEVEL ---ntime=0:00.01
DATA: TIME real time           5.21 seconds            720    WORK.PRVLEVEL ---ntime=0:05.21
PRNT: TIME real time           0.00 seconds            720    WORK.PRVLEVEL ---ntime=0:00.00
```

In addition, it puts them into a SAS data set for further analysis. (Figure 3.4)

Figure 3.4 Log Analysis Results Snapshot Three

	dsn	PROCDAT	obs	ctime	ntime
1	WORK.FACINFO	SORT:	6553	0.74	0.74
2	WORK.TEMP	SORT:	6554484	19.53	19.53
3	SAFLIB.M5_PLIST_201607	DATA:	6554484	27.00	27
4	SAFLIB.M5_PLIST_201607	SORT:	6554484	18.59	18.59
5	WORK.M5_PTLIST	DATA:	6554484	26.34	26.34
6	WORK.M5_PTLIST	SORT:	6554484	29.15	29.15
7	WORK.M5_PTLIST	SQL :	6554484	7.20	7.2
8	WORK.M5_PTLIST	DATA:	6554484	0.00	0
9	WORK.M5_PTLIST	SQL :	6554484	6.75	6.75

Then PROC MEANS summarizes the runtime of the whole process recorded in the SAS log file. As an example, the statistics of the M5 job for the 201607 run are shown below (Figure 3.5).

Figure 3.5 Log Analysis Results

PROCDAT	dsn	N Obs	Variable	Mean	Maximum	Minimum	Sum
DATA:	SAFLIB.M5_PLIST_201607	1	ntime	27.0000000	27.0000000	27.0000000	27.0000000
			obs	6554484.00	6554484.00	6554484.00	6554484.00
	WORK.M5_PTLIST	2	ntime	13.1700000	26.3400000	0	26.3400000
			obs	6554484.00	6554484.00	6554484.00	13108968.00
	WORK.PRVLEVEL	6426	ntime	6.1020090	20.9000000	5.1600000	39211.51
			obs	1019.99	6240.00	12.0000000	6554484.00
PRNT:	WORK.PRVLEVEL	6426	ntime	0.0066355	0.0700000	0	42.6400000
			obs	1019.99	6240.00	12.0000000	6554484.00
SORT:	SAFLIB.M5_PLIST_201607	1	ntime	18.5900000	18.5900000	18.5900000	18.5900000
			obs	6554484.00	6554484.00	6554484.00	6554484.00
	WORK.FACINFO	1	ntime	0.7400000	0.7400000	0.7400000	0.7400000
			obs	6553.00	6553.00	6553.00	6553.00
	WORK.M5_PTLIST	1	ntime	29.1500000	29.1500000	29.1500000	29.1500000
			obs	6554484.00	6554484.00	6554484.00	6554484.00
	WORK.TEMP	1	ntime	19.5300000	19.5300000	19.5300000	19.5300000
			obs	6554484.00	6554484.00	6554484.00	6554484.00
SQL :	WORK.M5_PTLIST	2	ntime	6.9750000	7.2000000	6.7500000	13.9500000
			obs	6554484.00	6554484.00	6554484.00	13108968.00

There are 6,428 DATA steps, 3 large ones, and 6,426 small ones. The large data steps only took a few minutes. And the 6,426 small data steps took more than 10 hours: 39,211/(60*60) seconds =10.89 hours. The PROC steps took less than a minute.

Based on the statistics shown above, we can tell that the %MACRO/%DO structure is very time consuming in this application. It posts an efficiency issue. In the next section, we will show the redevelopment of this application to make it more efficient.

IV. REDEVELOPING THE APPLICATION

Once we have identified the cause of the long runtime, we can redesign the application with efficiency in mind.

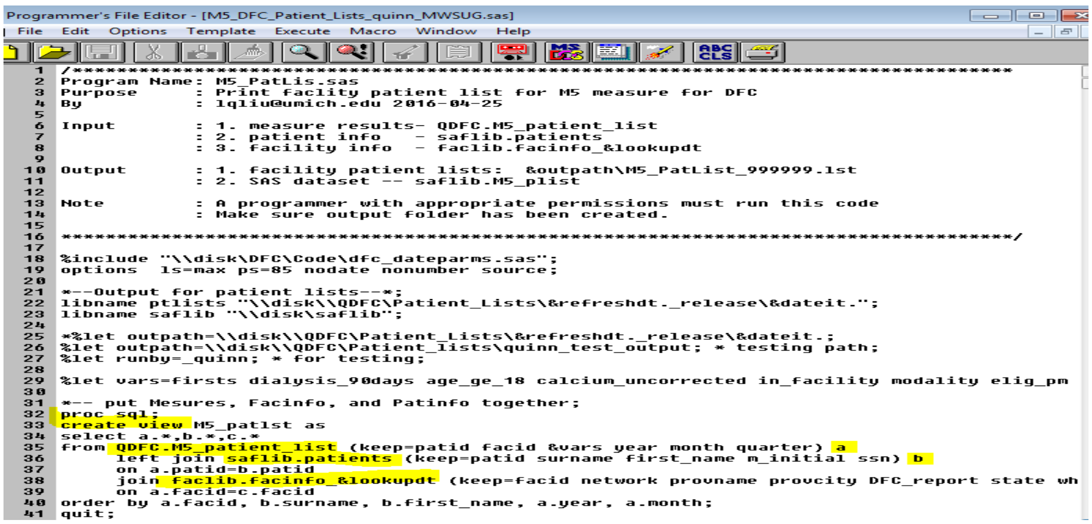
The first area to improve the original SAS application is to reduce the number of steps. Some data steps and procs can be combined, some steps and the %macro and data sorting can be avoided. SAS view can be used to replace data set. In addition, we can reduce the size of the log file by getting rid of macro related lines and fixing invalid data errors. That will make the log file more readable and save some I/O time as well. Second, and most importantly, for the reporting part, we can use a simple but powerful technique to avoid the 6,000+ small data steps: We use the SAS BY processing mechanism and DATA step **FILE statement** instead of the loops of DATA steps and PROC PRINTs.

Here is the outline of the re-developed SAS application. The new code only contains one PROC SQL view and one DATA step. There is no %macro/PROC PRINT/SORT. It uses a DATA step FILE statement with option FILEVAR= to write out facility specific reports.

```
PROC SQL; CREATE VIEW … AS …; QUIT;

DATA …;
  SET …;
  BY FACID;
  … …
  _FN= … FACID …;
  FILE WRITEOUT FILEVAR=_FN …;
  … …
  PUT …;
  … …
RUN;
```

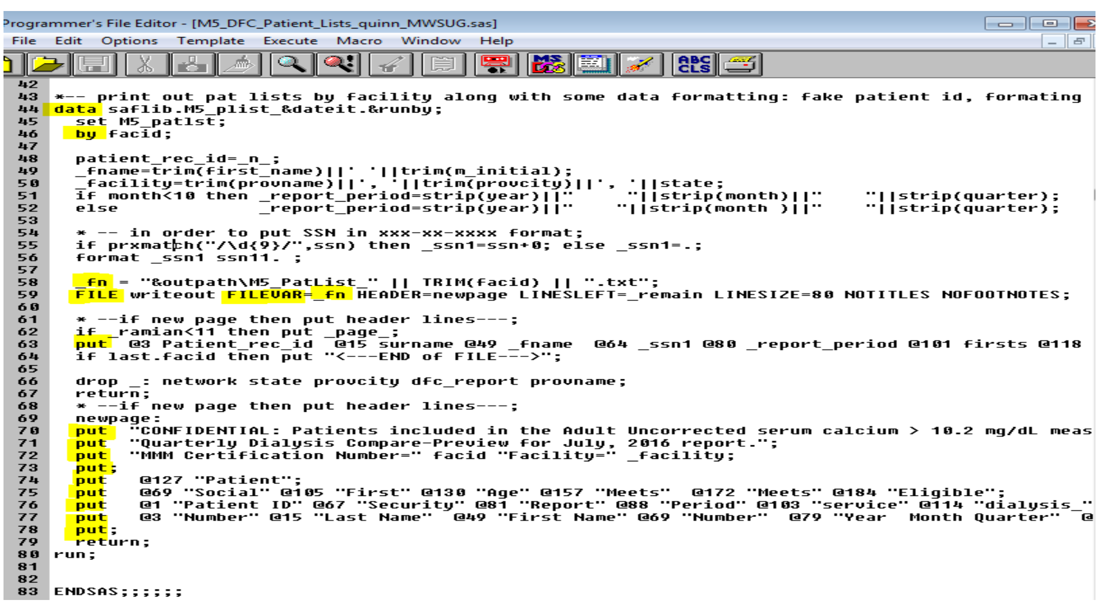
The new SAS application has only about 80 lines. (The original one has about 150 lines.)

Figure 4.1a Redeveloped Code (part 1)



```
/*****************************************************************************
Program Name: M5_PatLis.sas
Purpose      : Print faclity patient list for M5 measure for DFC
By           : lqliu@umich.edu 2016-04-25

Input        : 1. measure results- QDFC.M5_patient_list
             : 2. patient info   - saflib.patients
             : 3. facility info  - faclib.facinfo_&lookupdt

Output       : 1. facility patient lists:  &outpath\M5_PatList_999999.lst
             : 2. SAS dataset -- saflib.M5_plist

Note         : A programmer with appropriate permissions must run this code
             : Make sure output folder has been created.

*****************************************************************************/

%include "\\disk\DFC\Code\dfc_dateparms.sas";
options  ls=max ps=85 nodate nonumber source;

*--Output for patient lists--*;
libname ptlists "\\disk\\QDFC\Patient_Lists\&refreshdt._release\&dateit.";
libname saflib "\\disk\saflib";

*%let outpath=\\disk\\QDFC\Patient_Lists\&refreshdt._release\&dateit.;
%let outpath=\\disk\\QDFC\Patient_lists\quinn_test_output; * testing path;
%let runby=_quinn; * for testing;

%let vars=firsts dialysis_90days age_ge_18 calcium_uncorrected in_facility modality elig_pm

*-- put Mesures, Facinfo, and Patinfo together;
proc sql;
create view M5_patlst as
select a.*,b.*,c.*
from QDFC.M5_patient_list (keep=patid facid &vars year month quarter) a
     left join saflib.patients (keep=patid surname first_name m_initial ssn) b
     on a.patid=b.patid
     join faclib.facinfo_&lookupdt (keep=facid network provname provcity DFC_report state wh
     on a.facid=c.facid
order by a.facid, b.surname, b.first_name, a.year, a.month;
quit;
```

Figure 4.1b Redeveloped Code (part2)



```
*-- print out pat lists by facility along with some data formatting: fake patient id, formating
data saflib.M5_plist_&dateit.&runby;
  set M5_patlst;
  by facid;

  patient_rec_id=_n_;
  _fname=trim(first_name)||' '||trim(m_initial);
  _facility=trim(provname)||', '||trim(provcity)||', '||state;
  if month<10 then _report_period=strip(year)||"     "||strip(month)||"    "||strip(quarter);
  else             _report_period=strip(year)||"    "||strip(month )||"    "||strip(quarter);

  * -- in order to put SSN in xxx-xx-xxxx format;
  if prxmatch("/\d{9}/",ssn) then _ssn1=ssn+0; else _ssn1=.;
  format _ssn1 ssn11. ;

  _fn = "&outpath\M5_PatList_" || TRIM(facid) || ".txt";
  FILE writeout FILEVAR=_fn HEADER=newpage LINESLEFT=_remain LINESIZE=80 NOTITLES NOFOOTNOTES;

  * --if new page then put header lines---;
  if _ramian<11 then put _page_;
  put  @3 Patient_rec_id  @15 surname @49 _fname  @64 _ssn1 @80 _report_period @101 firsts @118
  if last.facid then put "<---END of FILE--->";

  drop _: network state provcity dfc_report provname;
  return;
  * --if new page then put header lines---;
  newpage:
  put  "CONFIDENTIAL: Patients included in the Adult Uncorrected serum calcium > 10.2 mg/dL meas
  put  "Quarterly Dialysis Compare-Preview for July, 2016 report.";
  put  "MMM Certification Number=" facid "Facility=" _facility;
  put;
  put   @127 "Patient";
  put   @69 "Social" @105 "First" @130 "Age" @157 "Meets"  @172 "Meets" @184 "Eligible";
  put   @1 "Patient ID" @67 "Security" @81 "Report" @88 "Period" @103 "service" @114 "dialysis_"
  put   @3 "Number" @15 "Last Name"  @49 "First Name" @69 "Number"  @79 "Year  Month Quarter"  @
  put;
  return;
run;


ENDSAS;;;;;;
```

The key SAS features used in the new application is FILE statement and its option FILEVAR=.

FILEVAR=variable

defines a variable whose change in value causes the FILE statement to close the current output file and open a new one the next time the FILE statement executes. The next PUT statement that executes writes to the new file that is specified as the value of the FILEVAR= variable.

Restriction:	The value of a FILEVAR= variable is expressed as a character string that contains a physical filename.
Interaction:	When you use the FILEVAR= option, the **file-specification** is just a placeholder, not an actual filename or a fileref that has been previously assigned to a file. SAS uses this placeholder for reporting processing information to the SAS log. It must conform to the same rules as a fileref.

The new SAS application produces the same results much more efficiently. Moreover, the log file (Figure 4.2) is nice and clean. It lists all the output files orderly. The runtime is 2.25 minutes (Figure 4.3). Can you believe it? The new application reduced the process time from about 10 hours to about 2 minutes. Comparing to the original version, it saved 99.6% of the runtime.

Figure 4.2 New Log Snapshot One

```
I:\MWSUG16__BB18\M5_DFC_Patient_Lists_quinn_MWSUG.log
NOTE: The file WRITEOUT is:
      Filename=\\DISK\quinn_test_output\MBD_PatList_111111.txt,
      RECFM=V,LRECL=256,File Size (bytes)=0,
      Last Modified=28Apr2016:20:52:01,
      Create Time=28Apr2016:16:29:03

NOTE: The file WRITEOUT is:
      Filename=\\DISK\\quinn_test_output\MBD_PatList_222222.txt,
      RECFM=V,LRECL=256,File Size (bytes)=0,
      Last Modified=28Apr2016:20:52:01,
      Create Time=28Apr2016:16:29:03
```

Figure 4.3 New Log Snapshot Two

```
I:\MWSUG16__BB18\M5_DFC_Patient_Lists_quinn_MWSUG.log
NOTE: The data set SAFLIB.M5_PLIST_201607_QUINN has 6554484 observations and 19 variables.
NOTE: Compressing data set SAFKECC.M5_PLIST_201607_QUINN decreased size by 42.20 percent.
      Compressed is 86102 pages; un-compressed would require 148966 pages.
NOTE: DATA statement used (Total process time):
      real time           2:02.22
      cpu time            1:45.83



196
197
198         ENDSAS;;;;;;

NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time           2:24.67
      cpu time            1:46.53
```

V. CONCLUSIONS

This real case study shows us that programming with efficiency in mind can make a great difference:

- 79 lines vs. 150 lines
- 1 step vs. 6,384 steps
- 22,518,989 vs. 61,852,446 records processed
- 00:02:30 vs. 11:02:17 (hh:mm:ss). Process time saved 99.62%.

Besides **raising awareness** for programming efficiency and introducing a **log analysis utility**, this case study presented two important suggestions to promote the performance of SAS applications.

First, developing a better SAS application requires a better understanding of the problem the application is to solve; once the problem is well understood, the programmer's problem solving skills help to design the right algorithm to tackle the problem. This design phase should involve as many knowledge and skills as possible, such as analytics, modular and parallel, data structure, logic/abstract/model and system thinking, etc.

Second, the application developer/programmer's SAS knowledge, experience, and skills also play an important role in programming efficiency. Here are some general SAS programming tips that can be usefully to improve application performance: use as fewer steps as possible if applicable; combine steps/remove unnecessary steps; process only the required variables and observations; avoid complex macro if you can; use simple/non-macro coding effective techniques; do not fall in love with your "hammer", know and pick the right tool to use; be machine, human and computing environment friendly.

REFERENCES

SAS Online Documentations for SAS 9.2, 9.3 and 9.4. (http://support.sas.com/documentation)

ACKNOWLEDGEMENTS

I would like to thank my colleagues at UM-KECC for their support. To name a few: Dr. Thomas Zheng reviewed and helped present some materials of this paper at a KECC journal club meeting in April 2016. Ms. Anca Tilea, Ms. Mia Wang and Dr. Thomas Zheng helped organize KECC journal club. Ms. Robin Padilla, Ms. Karen Wisniewski, Ms. Yating Sun and Ms. Natalie Scholz helped test the redeveloped application in May 2016. Ms. Megan Turf helped test the log analysis utility. Without the support from the KECC management team (Ms. Tempie Shearon, Ms. Valarie Ashby, Mr. Jas Sokhal, Ms. Casey Parrotte, Ms. Sally Sivrais and Dr. Joe Messana), it would not have been possible for this paper to be presented at the MWSUG conference. Jas and Valarie reviewed the draft slides. Ms. Susan Reimann directly helped with travel arrangement.

I also want to thank the MWSUG conference 2106 team, especially co-chairs of Beyond the Basic SAS, Ms. Melissa Ullman and Ms. Andrea Frazier, for answering my email requests when I was traveling overseas in August 2016.

CONTACT INFORMATION

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

Lingqun Liu
University of Michigan
Kidney Epidemiology and Cost Center
1415 Washington Heights, Suite 3645 SPH I
Ann Arbor, MI 48109-2029
Email: lqliu@umich.edu
http://www.kecc.sph.umich.edu/

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.