PhUSE 2010

Paper CS05

For a programming more efficient

Claude Guyot, Sanofi-Aventis, Chilly-Mazarin, France

ABSTRACT

We all use SAS® programs that run since years and years without updating them because they give the expected results. Nevertheless, we have to provide deliverables more and more quickly and we work on more and more big databases. So, it seems very useful and necessary to revise regularly the code to optimize the process of the programs to gain in performance and efficiency.

These revisions allow to take into account the evolutions of SAS® and the evolution of our own experience. The comparison of the statements, functions or procedures with equivalents in the result, can lead to a program more efficient and less consumer of CPU.

From an existing program and after a benchmark of the code, it will be interesting to see how this program can be improved and to see concretely the real gain.

INTRODUCTION

The topic of this presentation is not to revolutionize the SAS® programming but just to give some simple tricks to either gain time of programming, time of process or space on the disks.

REASONS FOR A PROGRAMMING MORE EFFICIENT

The requirements of the Health authorities force the laboratories to be ready to provide results or information on recent studies but also on old studies. So, data are kept more and more for a long time on production servers to be reachable easily.

Moreover, the efficacy or safety analyses on pool of studies are very disk space consumer.

The new standards imposed by the Health authorities multiply the number of data bases.

For all these reasons and for others, the space disk can become a concern very quickly for the laboratories.

Moreover, the aggressive deadlines imposed by the authorities, by the laboratories for CROs or simply by the hierarchy, force to be efficient in term of speed.

Of course, all these problems could be resolved technically with material more and more powerful with servers having big space capacity and network ultra efficient. But, in the reality, we know that we have to work with all these issues during a time more and less long. So, by waiting, we have to take habits to compensate technical lacks by, for example, improve the SAS® programming with simple rules.

SAVE THE TIME OF PROGRAMMING

A PROGRAM MUST BE CLEAR AND READABLE

A program is not just a succession of statements, data steps or procedures in only one block. All the different steps must be well dissociated by changing lines for each new statement, by skipping lines to differentiate clearly all the parts, by using the indentation to see clearly the beginning and the end of a data step, of a procedure, of a DO loop…

If this programming rule is not so useful for the writer of the program, it will be for the other people who will use the program and will maintain it.

PROGRAMS MUST BE WELL DOCUMENTED

To make a program accessible by other programmers and comprehensible in the time, a program must be well documented to understand what it is done and why, to follow the different changes, corrections, updating.

A header must explain the goal of the program, the pre-requisites neceesary, the history of the program (the nature with the authors and dates)

The comments inside the code serve to understand the different steps of the program so to follow the algorythm. Sometimes, it is easier to write a complete documentation to explain complex programs as standard macro-programs.

SAVE YOUR TIME AND THE TIME OF THE OTHERS

There are a lot of tricks to avoid unnecessary processing and to gain CPU time

SELECTION OF VARIABLES

A simple action to gain CPU time and space is to keep only what it is useful for the aim of the program. So, according to what it is easier, it is recommended to use the KEEP or DROP statements in the data steps and in the procedures to reduce the quantity of data to manage.

THE WHERE CLAUSE INSTEAD OF IF CONDITION

In a DATA step, the WHERE claude allows selecting records without reading all the records of the dataset contrary to the IF statement

In the procedures, the WHERE clause allows to select the records to keep.

CHOICE THE BETTER WAY TO JOIN 2 DATASETS

EXAMPLE: SELECT THE RECORDS OF A DATASET ACCORDING TO A SELECTION DONE IN A FIRST DATASET

With a merge

```
proc sort data=rdata.PATIENT1 (keep=PATID SEX) out=PATIENT2 (drop=SEX) ;
  by PATID ;
  where SEX='1' ;
run ;

proc sort data=rdata.THER out=THER1 ;
  by PATID ;
run ;

data PATTHER_MERGE ;
  merge PATIENT2 (in=a) THER1 (in=b) ;
  by PATID ;
  if a and b ;
run ;
```

With SQL

```
proc sql ;
  create table PATTHER_SQL (drop=_PATID)
  as select a.PATID, b.*
  from rdata.PATIENT1 (keep=PATID SEX) as a, rdata.THER
(rename=(PATID=_PATID)) as b
  where a.PATID=b._PATID and a.SEX='1' ;
quit ;
```

With format

```
data PATID_COD (keep=fmtname start end label) ;
  set rdata.PATIENT1 end=fin ;

  if sex='1' then do ;
    fmtname="$pat_cod" ;
    start=PATID ;
    end=PATID ;
```

```
    label=PATID ;
    output ;
  end ;

  else do ;
    fmtname="$pat_cod" ;
    start=PATID ;
    end=PATID ;
    label="Other" ;
    output ;
  end ;
run ;

proc format library=work.formats cntlin=work.PATID_COD ;
quit ;

data PATTHER_FORMAT ;
  set rdata.ther ;
  where put(PATID, $pat_cod.) ne "Other" ;
run ;
```

With a macro-variable

```
proc sql noprint ;
  select quote(PATID) into :patid_cod separated by ','
  from rdata.PATIENT1
  where sex="1" ;
quit ;

data PATTHER_MACROVAR ;
  set rdata.ther ;
  where PATID in (&patid_cod) ;
run ;
```

With the Hash method

```
proc sql noprint ;
  select NAME into :var_pat separated by ' '
  from sashelp.vcolumn
  where LIBNAME="RDATA" and MEMNAME="PATIENT1" ;

  select quote(strip(NAME)) into :var_the separated by ','
  from sashelp.vcolumn
  where LIBNAME="RDATA" and MEMNAME="THER" ;
quit ;

data PATTHER_HASH ;
  declare hash tab_hash() ;
  rc=tab_hash.DefineKey("PATID") ;
  rc =tab_hash.DefineData (&var_the);
  rc=tab_hash.DefineDone() ;

  do until (eof1) ;
    set rdata.PATIENT1 (where=(SEX='1')) end=eof1 ;
    rc=tab_hash.add() ;
    drop  &var_pat ;
  end ;

  do until (eof2) ;
```

```
    set rdata.THER end=eof2 ;
    if tab_hash.find()=0 then output;
  end ;
run ;
```

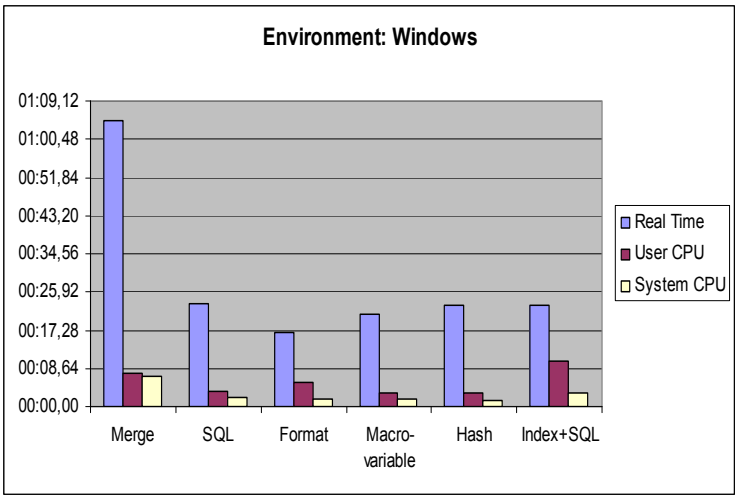
With index

```
proc sql ;
  create index PATID on rdata.PATIENT1 (PATID);
  create index PATID on rdata.THER (PATID);

  create table PATTHER_INDEX (drop=SEX _PATID)
  as select a.*, b.*
  from rdata.PATIENT1 (keep=PATID SEX) as a, rdata.THER
(rename=(PATID=_PATID)) as b
  where a.PATID=b._PATID and a.SEX='1' ;
quit ;
```

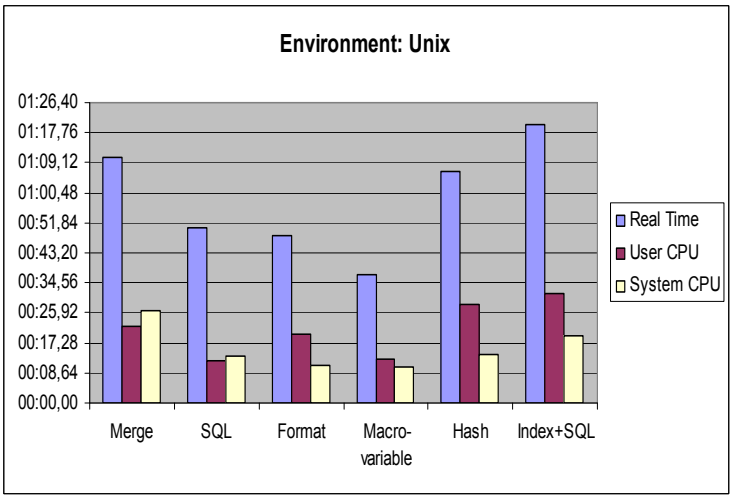
Performances of all methods: Windows

Environment: WINDOWS	Steps	Real Time	User CPU	System CPU
Merge	1st SORT	00:01,75	00:00,00	00:00,04
	2nd SORT	00:52,60	00:05,21	00:05,31
	Merge	00:10,21	00:02,16	00:01,62
	Total	**01:04,56**	**00:07,37**	**00:06,97**
SQL		**00:23,27**	**00:03,32**	**00:01,95**
Format	Data step	00:00,06	00:00,01	00:00,03
	PROC FORMAT	00:00,36	00:00,03	00:00,03
	Data step	00:16,35	00:05,56	00:01,55
	Total	**00:16,73**	**00:05,61**	**00:01,60**
Macro-variable	PROC SQL	00:02,32	00:00,01	00:00,01
	Data step	00:18,44	00:03,06	00:01,83
	Total	**00:20,76**	**00:03,07**	**00:01,84**
Hash	PROC SQL	00:00,51	00:00,00	00:00,01
	Data step	00:22,52	00:03,18	00:01,49
	Total	**00:23,03**	**00:03,18**	**00:01,50**
Index+SQL		**00:22,88**	**00:10,40**	**00:03,15**



Performances of all methods: Unix

Environment: Unix	Steps	Real Time	User CPU	System CPU
Merge	1st SORT	00:01,27	00:00,74	00:00,25
	2nd SORT	00:36,34	00:13,26	00:17,09
	Merge	00:33,20	00:08,25	00:09,43
	Total	**01:10,81**	**00:22,25**	**00:26,77**
SQL		**00:50,33**	**00:12,01**	**00:13,55**
Format	Data step	00:01,25	00:00,02	00:00,25
	PROC FORMAT	00:00,23	00:00,09	00:00,04
	Data step	00:46,54	00:19,91	00:10,50
	Total	**00:48,01**	**00:20,02**	**00:10,79**
Macro-variable	PROC SQL	00:01,16	00:00,03	00:00,18
	Data step	00:35,89	00:12,73	00:10,18
	Total	**00:37,05**	**00:12,76**	**00:10,36**
Hash	PROC SQL	00:00,11	00:00,01	00:00,02
	Data step	01:06,50	00:28,23	00:13,87
	Total	**01:06,60**	**00:28,24**	**00:13,90**
Index+SQL		**01:20,29**	**00:31,42**	**00:19,45**



SAVE THE SPACE

A recurrent topic is the lack of disk space due to more and more data. So, what is possible to do on the datasets to reduce their size?

ADAPT THE VARIABLES LENGTH

Not too small to not truncate the information
Not too large to not take space unnecessarily
Default length of the numeric variables is 8 but, according to the kind of information, can be less

THE SAS® DATASETS COMPRESSION

The dataset option COMPRESS=YES allows to compress the character variables. But this method can not be used in all cases. It depends of the structure: a dataset with mainly numeric variables must not be compressed because the size could increase instead of decrease.

CONCLUSION

The optimization of the programs is a necessity by it can be done carefully according to the data, to the priorities of the bussiness and to the technology already in place

CONTACT INFORMATION

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

Claude Guyot
Sanofi-Aventis
1, avenue Pierre Brossolette
Chilly-Mazarin / 91385
Work Phone: 01.60.49.72.72
Email: claude.guyot@sanofi-aventis.com