Hash Object Working Efficiently with Large SAS Datasets

Yazhen Wang, Fountain Medical Development Co., Ltd
Hao Xu, Fountain Medical Development Co., Ltd

PharmaSUG China 2016
Paper #88

Outline

- Introduction
- Hash object
- Conclusions

Section 1 : Introduction

Large SAS dataset

- Multiple center, Large sample clinical trial may generate large data
- Labs data
- Study Integration

What's the problem

- Time-consuming process in SAS
  - up to an hour cost for one program
- Engross your disk space
  - make public space crowded, it will also have a bad impact on others' work

Some solutions

1. Do not sort again for the sorted datasets

```
Data xx;
      set <dataset>;
      by var1 var2 …;
Run;
```

2. Use Class statement in PROC step rather than By statement
3. Use WHERE statement instead of IF statement
4. Remove unused variables
5. Proc delete

Section 2 : Hash Object

Example 1　Join large dataset (without sorting beforehand)

Abstract treatment group values of each subject from ADSL(subject-level analysis dataset) and use this information to derive laboratory analysis data set. The laboratory data set contains 183034 observations.

ADSL			
USUBJID	SEX	TRT01A	TRT01B
ABC-1001	Male	AB	AC
ABC-1004	Female	AC	AB
ABC-2013	Female	AB	AC
ABC-3002	Male	AC	AB
ABC-5017	Male	AB	AC
ABC-5022	Female	AC	AB
ABC-6011	Male	AB	AC

LB	
USUBJID	LBTESTCD
ABC-1001	ALB
ABC-1001	AST
ABC-1004	ALB
ABC-1004	AST
ABC-1004	HGB
ABC-2013	ALB
ABC-2013	AST

➢ Method A ----Standard Merge Statment

```
Data adlb;
  merge sds.lb (in=a)
        ads.adsl(in=b keep=crosssdt crossfl trt01p trt01a trtdur1 usubjid);
    by usubjid;
Run;
```

```
NOTE: There were 183034 observations read from the data set SDS.LB.
NOTE: There were 311 observations read from the data set ADS.ADSL.
NOTE: The data set WORK.ADLB has 183034 observations and 39 variables.
NOTE: DATA statement used (Total process time):
      real time           51.53 seconds
      cpu time            25.00 seconds
```

➢ Method B----Hash Object Method in a DATA Step

```
904   data adlb;
905           attrib crosssdt length=8
906                  crossfl length=$1
907                  trt01p length=$25
908                  trt01a length=$25
909                  trtdur1 length=8
910                  ;
911           if _n_=1 then do;
912           declare hash e(dataset: 'ads.adsl');
913                       e.definekey('usubjid');
914                       e.definedata('crosssdt','crossfl','trt01p','trt01a','trtdur1');
915                       e.definedone();
916
917                       call missing(crosssdt,crossfl,trt01p,trt01a,trtdur1);
918           end;
919           set sds.lb;
920           drop rc;
921           rc=e.find();
922   run;

NOTE: There were 311 observations read from the data set ADS.ADSL.
NOTE: There were 183034 observations read from the data set SDS.LB.
NOTE: The data set WORK.ADLB has 183034 observations and 39 variables.
NOTE: DATA statement used (Total process time):
      real time          39.45 seconds
      cpu time           17.76 seconds
```

Two methods comparison

- Merge processing time
- Real time 57.53s
- CPU time 25.00s

- Hash processing time
- Real time 39.45s
- CPU time 17.76s

Real time : -31%
CPU time : -29%

Point 1 : The hash object is more efficient to look up than the standard merge process

Standard Merge

Hash Object Method

Two methods summary

- Properly sorted before merge
- The merge key value should be consistent before merge

Standard Merge

- No need to sort
- Make the key value consistent in the DATA step

Hash Object Method

Example 2

Abstract arm values of each subject from SDTM.DM. The laboratory data set contains 184165 observations.

LAB2	
SUBJID	LBTEST
100022205	WBC
100022207	WBC
100022206	WBC

DM	
USUBJID	ACTARM
PERSIST-2-10002-2205	ABC
PERSIST-2-10002-2206	Placebo
PERSIST-2-10002-2207	Screen failure

Step 1: Create same variable and same value in two datasets

Step 2: Sort dataset LAB2 by key variable

Step 3: Merge two datasets by key variable

```
1360  data lab2;
1361       set lab2;
1362       usubjid = catx("-","PERSIST-2",substr( subjid,1,5),substr( subjid,6,4));
1363  run;

NOTE: There were 184165 observations read from the data set WORK.LAB2.
NOTE: The data set WORK.LAB2 has 184165 observations and 58 variables.
NOTE: DATA statement used (Total process time):
      real time           8.00 seconds
      cpu time            4.26 seconds


1364
1365  proc so
1366       by
1367  run;

NOTE: There w                                                    AB2.
NOTE: The data                                                   les.
NOTE: PROCEDU
      real ti
      cpu time            8.12 seconds


1368
1369  data test1;
1370       merge lab2 sdtm.dm(keep=usubjid actarmcd);
1371       by usubjid;
1372  run;

NOTE: There were 184165 observations read from the data set WORK.LAB2.
NOTE: There were 431 observations read from the data set SDTM.DM.
NOTE: The data set WORK.TEST1 has 184196 observations and 59 variables.
NOTE: DATA statement used (Total process time):
      real time           8.53 seconds
      cpu time            4.07 seconds
```

Real time: 38.18 seconds
CPU time: 16.45 seconds

Hash method: One Data step !

```
1732  data test(drop= rc);
1733          attrib actarmcd length=$8;
1734          if _n_=1 then do;
1735              declare hash demo (dataset: 'sdtm.dm');
1736                                  demo.definekey ('usubjid');
1737                                  demo.definedata ('actarmcd');
1738                                  demo.definedone ();
1739                                  call missing(actarmcd);
1740          end;
1741          set lab2 ;
1742          usubjid = catx("-","PERSIST-2",substr( subjid,1,5),substr( subjid,6,4));
1743          rc = demo.find();
1744  run;

NOTE: There were 431 observations read from the data set SDTM.DM.
NOTE: There were 184165 observations read from the data set WORK.LAB2.
NOTE: The data set WORK.TEST has 184165 observations and 59 variables.
NOTE: DATA statement used (Total process time):
      real time           8.18 seconds
      cpu time            6.70 seconds
```

Two methods comparison

- Merge processing time
- Real time 38.18s
- CPU time 16.45s

- Hash processing time
- Real time 8.18s
- CPU time 6.70s

Real time : -78.6%
CPU time : -59.3%

Point 2 : The hash object is more directly than the regular method

Standard Merge

Hash Object Method

Section 3: Conclusions

Conclusions

- High efficient algorithm to look up
- No need to sort
- Create key variable when join dataset in one DATA step
- Do well in some special case (refer to paper)

Reference

- SAS Hash Object Programming Made Easy, Michele M. Burlew, Russ Lavery

Thanks for your attention!

Name: Yazhen Wang
Organization: Fountain Medical Development Co., Ltd
Address:Room 403, Building 43, No.70, Headquarter Base,
Phoenix Road, Jiangning District
City, State ZIP: Nanjing, 210000
Work Phone: +86 18936015398
E-mail: yazhen.wang@fountain-med.com