EPS

Phuse Single Day Event —TOKYO 2019

Handling Big Data Using SAS
(HPDS2 , HASH object and FCMP)
--Do Not Sort! Reduce Steps!

December 10, 2019

EPS Corporation
Yutaka Morioka

Presenters

Yutaka Morioka (35)

Besides my work as a clinical data scientist, actively engaging in disseminating various SAS programming techniques from introductory level to advanced.

>General Biography

Drug Wholesaling[MS] –1 year

↓

CRO[DM,STAT] – Total 5 years

↓

Research Company – Total 2 years

↓

(2016-Now) EPS−STAT

>Presentation History

SAS Japan Users General Forum (2013～2019 – Total 10 Presentations)

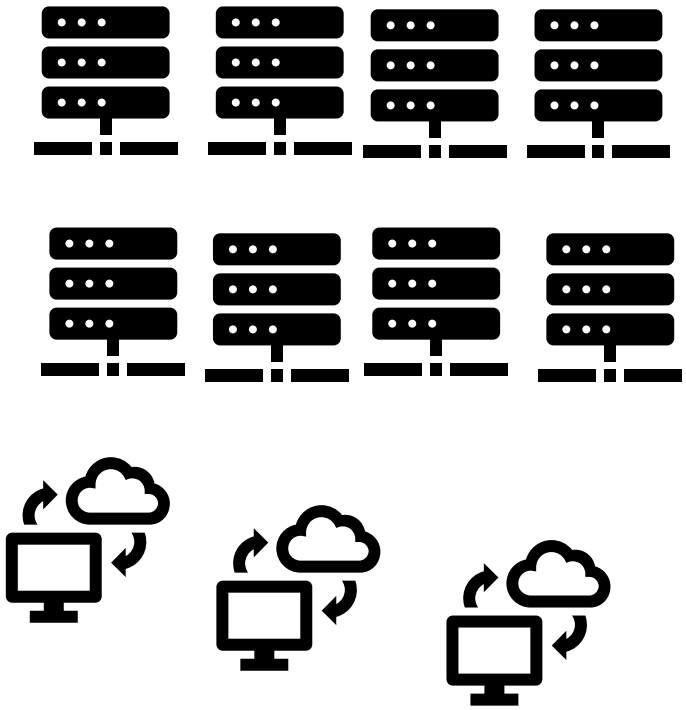
SAS Global Forum 2019

Pharma SUG Single Day Event – Tokyo 2019

Phuse Single Day Event – Tokyo 2019 ←Now

Is only the system environment important?

The topic of big data analysis tends to be only about architecture.

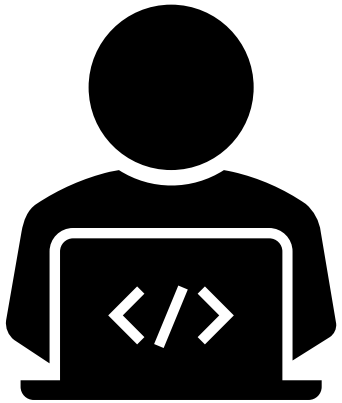


Do you make light of programming techniques?

System environment is the most important.

However, the technique is just as the same importance.

Today I'd like to talk about the technique of handling Big Data.



Excuse me, I disseminate SAS fanatically.

I can only talk about SAS maniacally.

Proc HPxxx

SAS® 9.4 SAS HIGH PERFORMANCE PROCEDURES

High Performance Statistics	High Performance Data Mining	High Performance Econometrics	High Performance Forecasting	High Performance Optimization	High Performance Text Mining
HPLOGISTIC HPGENSELECT HPREG HPLMIXED HPNLMOD HPSPLIT HPFMM HPCANDISC HPPRINCOMP HPPLS HPQUANTSELECT	HPREDUCE HPNEURAL HPFOREST HP4SCORE HPDECIDE HPCLUS HPSVM HPBNET HPTSDR	HPCOUNTREG HPSEVERITY HPQLIM HPPANEL HPCOPULA HPCDM	HPFORECAST	OPTLSO Select features in OPTMILP OPTLP OPTMODEL OPTGRAPH HPCDM	HPTMINE HPTMSCORE
Common Set: HPDS2, HPDMDB, HPSAMPLE, HPSUMMARY, HPIMPUTE, HPBIN, HPCORR					

Prefix with hp

```
proc hplogistic data=sashelp.JunkMail;
   model Class(event='1')=Make Address All _3d Our Over Remove Internet Order
         Mail Receive Will People Report Addresses Free Business Email You
         Credit Your Font _000 Money HP HPL George _650 Lab Labs Telnet _857
         Data _415 _85 Technology _1999 Parts PM Direct CS Meeting Original
         Project RE Edu Table Conference Semicolon Paren Bracket Exclamation
         Dollar Pound CapAvg CapLong CapTotal;
run;
```

```
proc logistic data= sashelp. JunkMail;
   model Class(event='1')=Make Address All _3d Our Over Remove Internet Order
         Mail Receive Will People Report Addresses Free Business Email You
         Credit Your Font _000 Money HP HPL George _650 Lab Labs Telnet _857
         Data _415 _85 Technology _1999 Parts PM Direct CS Meeting Original
         Project RE Edu Table Conference Semicolon Paren Bracket Exclamation
         Dollar Pound CapAvg CapLong CapTotal;
run;
```

※ Syntax is similar

Trade-off

```
NOTE: The HPLOGISTIC procedure is executing in single-machine mode.
NOTE: You are modeling the probability that Class='1'.
NOTE: Convergence criterion (GCONV=1E-8) satisfied.
NOTE: There were 4601 observations read from the data set SASHELP.JUNKMAIL.
NOTE: PROCEDURE HPLOGISTIC used (Total process time):
      real time           0.07 seconds
      cpu time            0.17 seconds
```

```
NOTE: PROC LOGISTIC is modeling the probability that Class=1.
NOTE: Convergence criterion (GCONV=1E-8) satisfied.
NOTE: There were 4601 observations read from the data set SASHELP.JUNKMAIL.
NOTE: PROCEDURE LOGISTIC used (Total process time):
      real time           0.14 seconds
      cpu time            0.14 seconds
```

Real time  CPU time 

However, since it depends on the execution environment and data, it may be slow.

PERFORMANCE statement

```
proc hpsummary data=TEST;
        class X;
        var Y;
        performance nthreads = 3;
        output out=B;
run;
```

Easy threading♪

The PERFORMANCE statement defines performance parameters for multithreaded and distributed computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of a high-performance analytical procedure.

<options>
GRIDHOST=*"name"*
HOST=*"name"*
GRIDMODE=SYM | ASYM
MODE=SYM | ASYM
NODES=ALL | *n*
etc .. etc..

With proc DS2, SAS has introduced a significant alternative to the DATA step by introducing an object-oriented programming.

DS2 enable us to access, manage,and share data in a scalable, threaded, and standards-based way.

DS2 supports the following general object-oriented concepts:

- Object
- Instance
- Attribute
- Method

It means that general program development methods can also be applied to SAS.

DS2 Procedure

```
proc ds2 ;
data OUT12(overwrite=yes);
 declare package hash h1();
 declare double ID MEAN_VAL;
 drop mean;
  method init();
   h1.dataset('{select SEX ,mean(VAL) as
                MEAN_VAL
                from TEST group by SEX}');
   h1.keys([SEX]);
   h1.data([MEAN_VAL]);
   h1.definedone();
 end;
 method run();
  set TEST2;
  h1.find();
 end;
enddata;
run;
quit;
```

SQL and Hash collaboration

Similar to traditional data steps

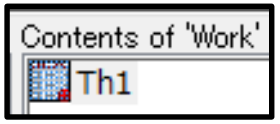
Overload

```
method cal_bsa(double h,double w) returns double;
 declare double result;
 result = w**0.425*h**0.725*0.007184;
 return result;
end;

method cal_bsa(char h,char w) returns double;
 declare double result;
 result =
inputn(w,'best.32')**0.425*inputn(h,'best32.')**0.725*0.007184;
 return result;
end;

method cal_bsa(double h,double w,char cm) returns double;
  declare double result;
  if cm ='Niitani' then result=w**0.425*h**0.725*0.007358;
  if cm ='Fujimoto'then result=w**0.444*h**0.663*0.008883;
return result;
end;
```

My Favorite♪

First, define the process you want to thread

```
proc ds2;
thread th1/overwrite=yes;
declare double i thread;
method run();
 do i = 1 to 10;
  thread=_threadid_;
  put _threadid_=;
  output;
 end;
end;
endthread;
run;
quit;
```



SAS_ROWID_	SAS_CHECKSUM_	SAS_TEXTTHREAD_
1	18122651414	DATA_STEP_2_PROGRAM
2	60987836285	1.0
3	24665124962	224
4	13200331321	U5YTEUF44AITQS2YV2ELJ6OLOV
5	66882918612	6SQELITTO7NHRTVGGWVNKALTZ

Create an instance and execute the process

```
proc ds2;
data OUT(overwrite=yes);
 declare thread th1 t;
 declare double NO;
 method run();
  set from t threads=3;
   NO=_N_;
  end;
 enddata;
run;
quit;
```

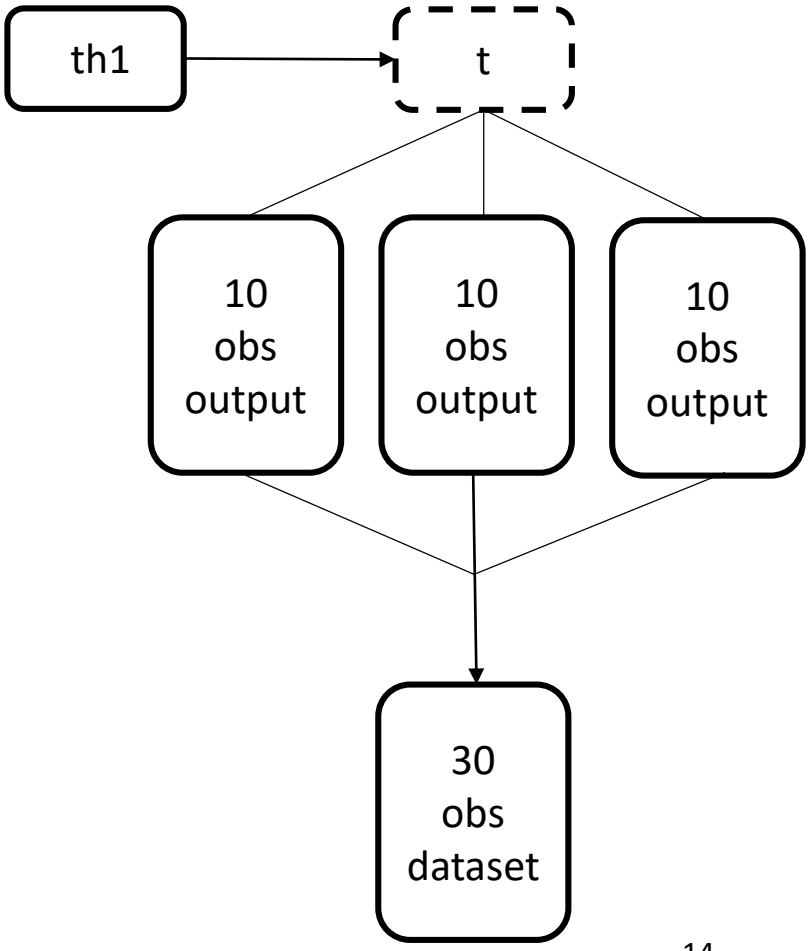
DS2-Threading(1)

LOG

```
65   proc ds2;
66   data OUT(overwrite=yes);
67    declare thread th1 t;
68    declare double NO;
69    method run();
70     set from t threads=3;
71      NO=_N_;
72     end;
73    enddata;
74   run;
_THREADID_=1
_THREADID_=1
_THREADID_=1
_THREADID_=1
_THREADID_=1
_THREADID_=1
_THREADID_=1
_THREADID_=1
_THREADID_=1
_THREADID_=1
_THREADID_=2
_THREADID_=2
_THREADID_=2
_THREADID_=2
_THREADID_=2
_THREADID_=2
_THREADID_=2
_THREADID_=2
_THREADID_=2
_THREADID_=3
_THREADID_=3
_THREADID_=3
_THREADID_=3
_THREADID_=3
_THREADID_=3
_THREADID_=3
_THREADID_=3
_THREADID_=3
_THREADID_=3
NOTE: Execution succeeded. 30 rows affected.
75   quit;

NOTE: PROCEDURE DS2 used (Total process time):
      real time           0.04 seconds
      cpu time            0.04 seconds
```

DATASET

NO	i	thread
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1
11	1	2
12	2	2
13	3	2
14	4	2
15	5	2
16	6	2
17	7	2
18	8	2
19	9	2
20	10	2
21	1	3
22	2	3
23	3	3
24	4	3
25	5	3
26	6	3
27	7	3
28	8	3
29	9	3
30	10	3



DS2-Threading(2)

```
data TEST;
array ar{100};
do i= 1 to 10000000;
  do j = 1 to 100;

ar{j}=rand('uniform');
  end;
  output;
 end;
 drop i j;
run;
```

Test data
-10 million obs
-100 variables

```
proc ds2;
thread th2/overwrite=yes;
 dcl double count threadid;
 drop count;
 method run();
  set TEST;
  count +1;
  threadid=_threadid_;
 end;
 method term();
  put '_threadid_=' _threadid_ '
reads ' count 'obs';
 end;
endthread;
 data OUT(overwrite=yes);
  dcl thread th2 t;
  method run();
   set from t threads=3;
  end;
 enddata;
run;
quit;
```

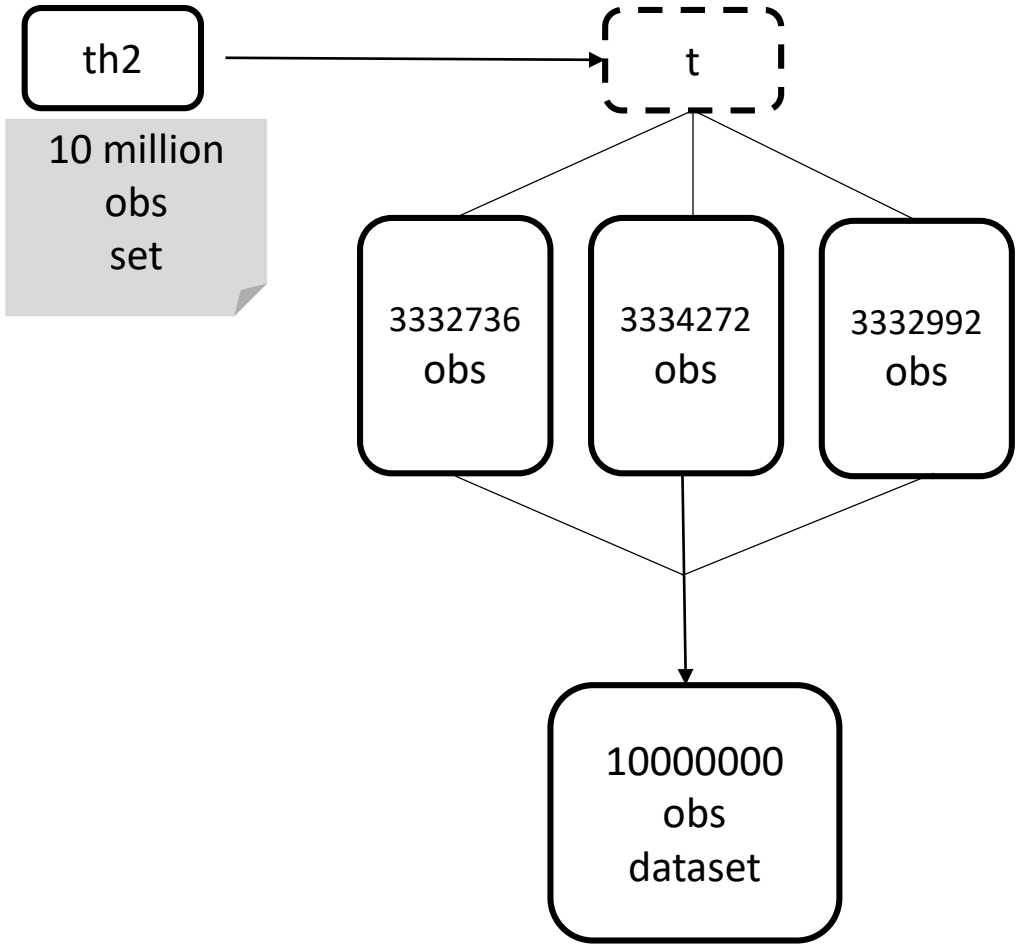
define

execute

DS2-Threading(2)

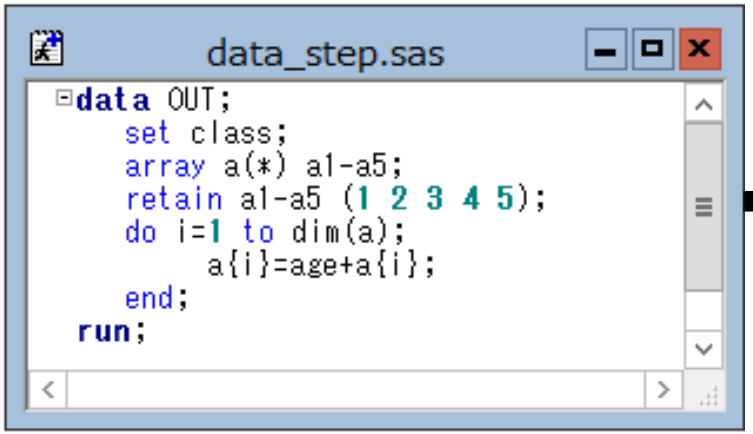
LOG

```
280  proc ds2;
281  thread th2/overwrite=yes;
282   dcl double count threadid;
283   drop count;
284   method run();
285    set TEST;
286    count +1;
287    threadid=_threadid_;
288   end;
289   method term();
290    put '_threadid_=' _threadid_ ' reads ' count 'obs';
291   end;
292  endthread;
293
294   data OUT(overwrite=yes);
295    dcl thread th2 t;
296    method run();
297     set from t threads=3;
298    end;
299   enddata;
300  run;
_threadid_= 2  reads  3332736 obs
_threadid_= 1  reads  3334272 obs
_threadid_= 3  reads  3332992 obs
NOTE: Created thread th2 in data set work.th2.
NOTE: Execution succeeded. 10000000 rows affected.
301
302  quit;

NOTE: PROCEDURE DS2 used (Total process time):
      real time           34.23 seconds
      cpu time            16.89 seconds
```

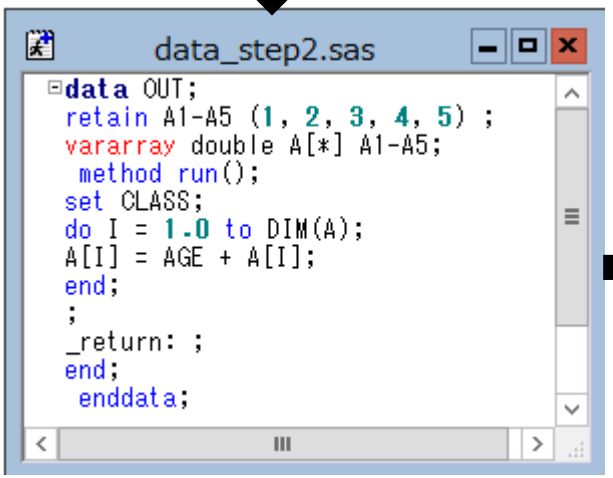


DSTODS2 Procedure

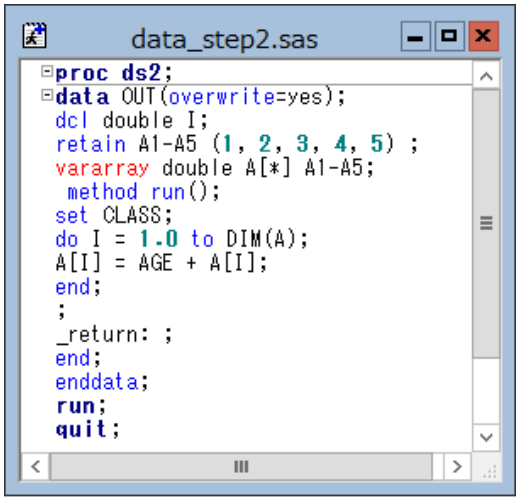
The DSTODS2 procedure enables you to translate a subset of your SAS DATA step code into DS2 code. Then you can revise your program to take advantage of DS2 features and submit your program using PROC DS2.




```
proc dstods2
 in="data_step.sas"
 out="data_step2.sas";
run;
```



manual modify



The HPDS2 procedure enables you to submit DS2 language statements from a Base SAS session to one or more machines in a grid for parallel execution.

```
proc hpds2 data=work.test out=work.out;
   data DS2GTF.out;
      method run();
         set DS2GTF.in;
      end;
   enddata;
run;
quit;
```

Keyword
"data=" links "DS2GTF.in"
"out=" links "DS2GTF.out"

The HPDS2 Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Auto Threading

Data Access Information			
Data	Engine	Role	Path
WORK.TEST	V9	Input	On Client
WORK.OUT	V9	Output	On Client

If you use HPDS2, you can use Perfomance Statement like proc HPXX;

```
proc hpds2 data=work.test out=work.out;
performance nthreads=8;
    data DS2GTF.out;
       method run();
          set DS2GTF.in;
       end;
    enddata;
run;
quit;
```

The SAS System

The HPDS2 Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	8

Data Access Information			
Data	Engine	Role	Path
WORK.TEST	V9	Input	On Client
WORK.OUT	V9	Output	On Client

Real time ↘ CPU time ↗

From now on, we will would like to talk less about environmental dependency but technique

Principle is …

Innovate data extraction!

Do Not Sort!

Reduce Steps!

Key match

Dataset: A

ID	TPT	VAL
002	1	63
002	2	89
002	3	84
001	1	71
001	2	44
001	3	80
003	1	77
003	2	24
003	3	46
005	1	17
005	2	90
005	3	50
004	1	47
004	2	45
004	3	32

Big

Dataset: B

ID	AGE
003	35
001	24
002	33
004	40
005	60

Small

Dataset: C

ID	TPT	VAL	AGE
002	1	63	33
002	2	89	33
002	3	84	33
001	1	71	24
001	2	44	24
001	3	80	24
003	1	77	35
003	2	24	35
003	3	46	35
005	1	17	60
005	2	90	60
005	3	50	60
004	1	47	40
004	2	45	40
004	3	32	40

Key:ID

```
proc sort data = A ;
 by ID;
run;

proc sort data = B ;
 by ID;
run;

data C;
 merge A(in=ina)
       B;
 by ID;
 if ina;
run;
```

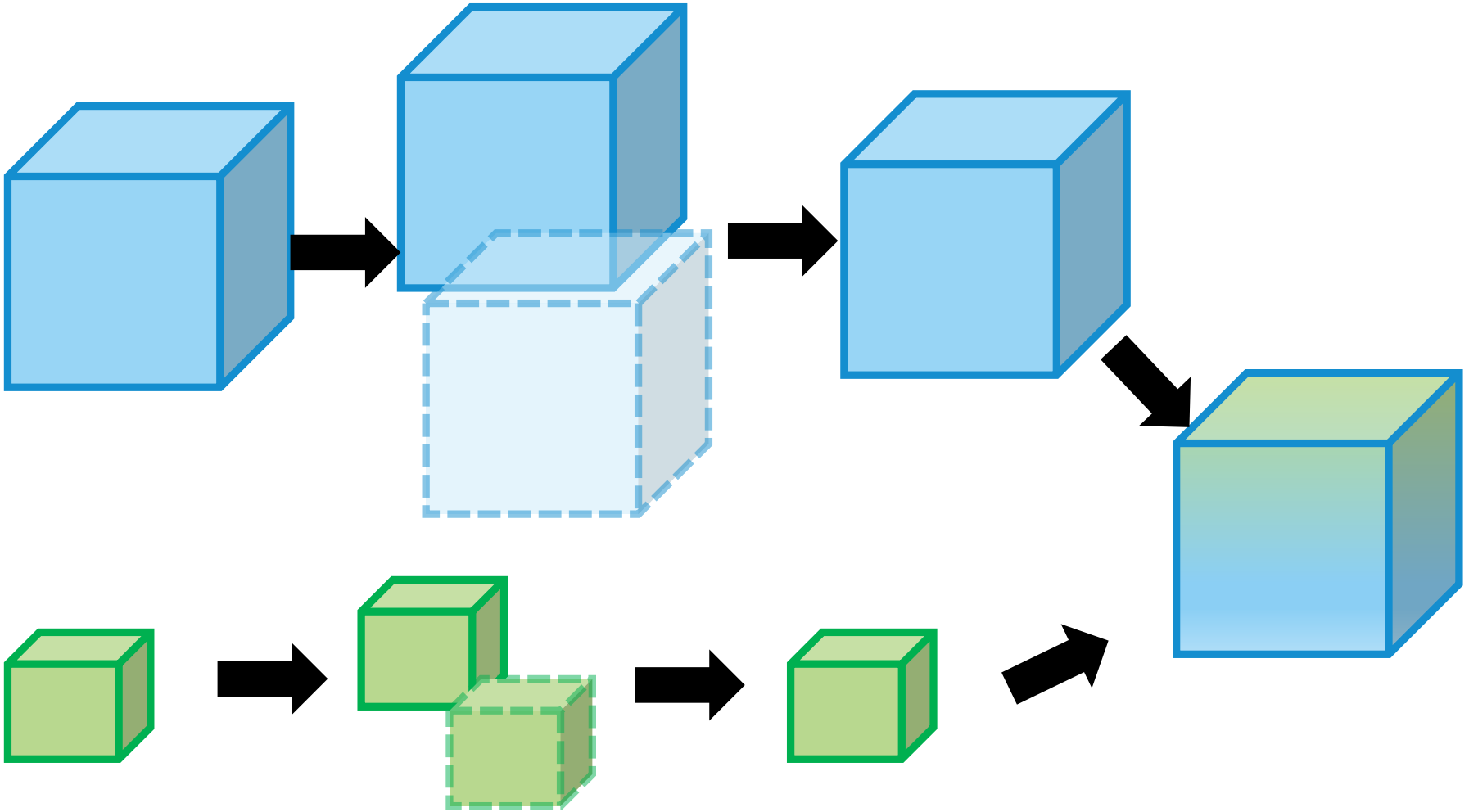
SORT

SORT

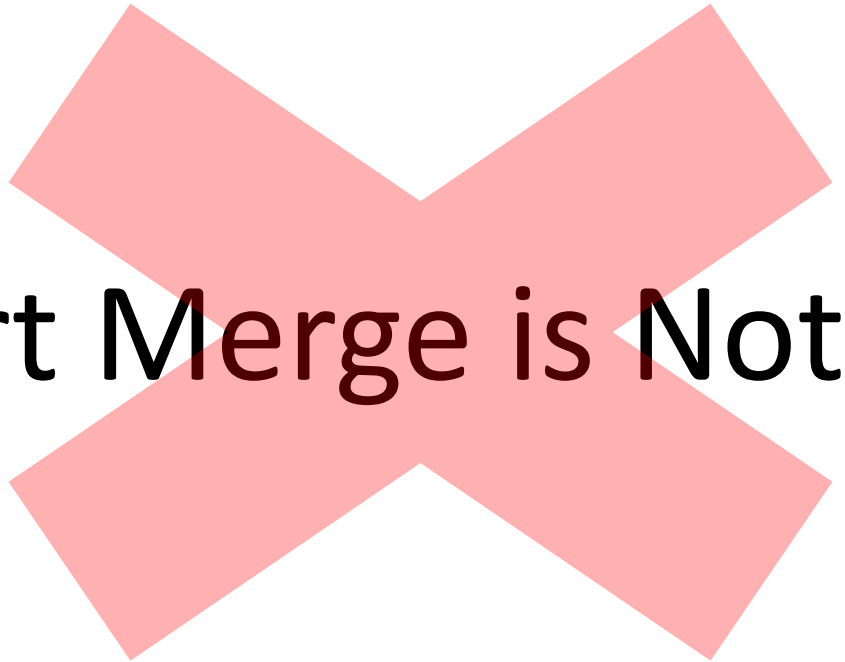
MERGE

3 steps (2 sort)

Proc sort almost doubles the required data area temporarily



Big table — Small master



Sort Sort Merge is Not Good!!

Proc sort Tips—Save resource

unstable sort

```
proc sort data= XX noequals;
```

overwrite

```
proc sort data= XX overwrite;
```

Don't forget to back up

tag sort

```
proc sort data= XX tagsort;
```

First sort by key, then process data

```
proc sql noprint;
   create table C as
   select A.ID,TPT,VAL,AGE
   from A left outer join B on A.ID=B.ID;
quit;
```

1 steps

However, if you increase the size of table……

```
ERROR: Sort execution failure.
```

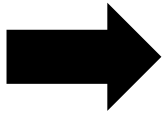
Sort ?? Why??

Dataset: A

ID	TPT	VAL
002	1	63
002	2	89
002	3	84
001	1	71
001	2	44
001	3	80
003	1	77
003	2	24
003	3	46
005	1	17
005	2	90
005	3	50
004	1	47
004	2	45
004	3	32

Dataset: B

ID	AGE
003	35
001	24
002	33
004	40
005	60



Dataset: C

ID	TPT	VAL	AGE
001	3	80	24
001	2	44	24
001	1	71	24
002	1	63	33
002	2	89	33
002	3	84	33
003	1	77	35
003	3	46	35
003	2	24	35
004	2	45	40
004	1	47	40
004	3	32	40
005	3	50	60
005	2	90	60
005	1	17	60

Sorted by ID

```
proc sql _tree;
   select A.ID,TPT,VAL,AGE
   from A left outer join B on
A.ID=B.ID;
quit;
```

```
 sqxslct
     sqxjm
         sqxsrc( WORK.B )
         sqxsort
             sqxsrc( WORK.A )
quit;
```

SQL join causes sorting internally

Big table — Small master

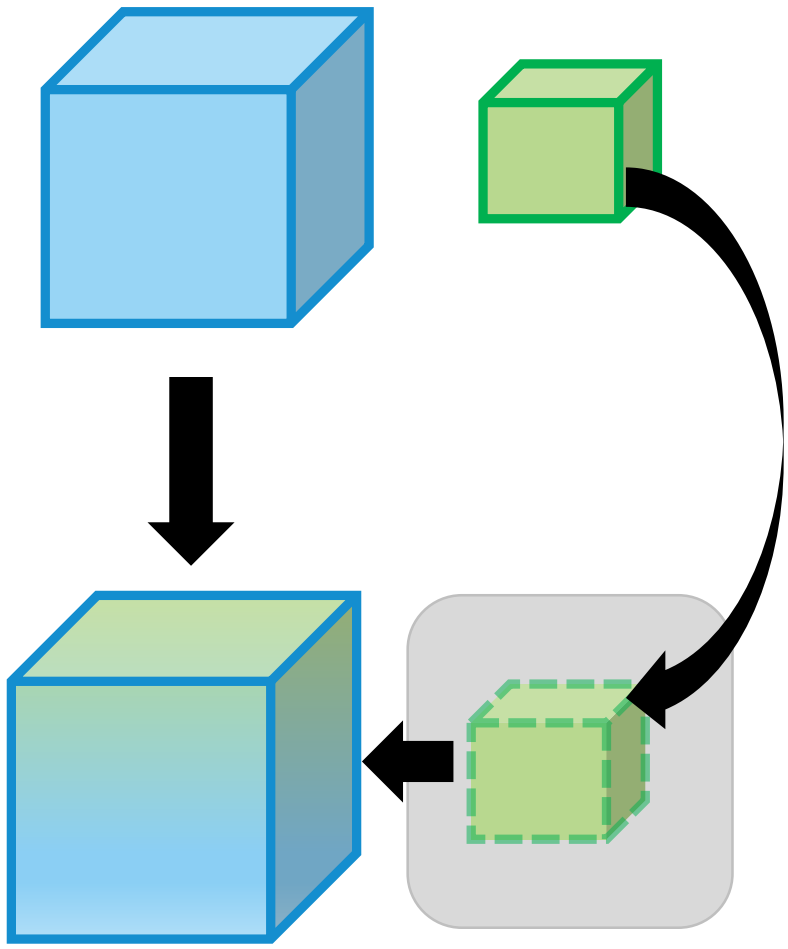
Hash Object

```
data C;
set A;
if 0 then set B;
if _N_=1 then do;
      declare hash h1(dataset:"B");
      h1.definekey("ID");
      h1.definedata("AGE");
      h1.definedone();
end;
if h1.find() then call missing(of AGE);
run;
```

Hash Object

```
data C;
set A;
if 0 then set B;
if _N_=1 then do;
        declare hash h1(dataset:"B");
        h1.definekey("ID");
        h1.definedata("AGE");
        h1.definedone();
end;
if h1.find() then call missing(of AGE);
run;
```

Hash object creates objects in memory and exchanges data with methods



Big table — Small master

Hash Object is the Best Choice!!

```
data C;
set A;
where ID in ("003","005");
run;
```

```
data C;
set A;
if ID in ("003","005");
run;
```

The result is the same, but…

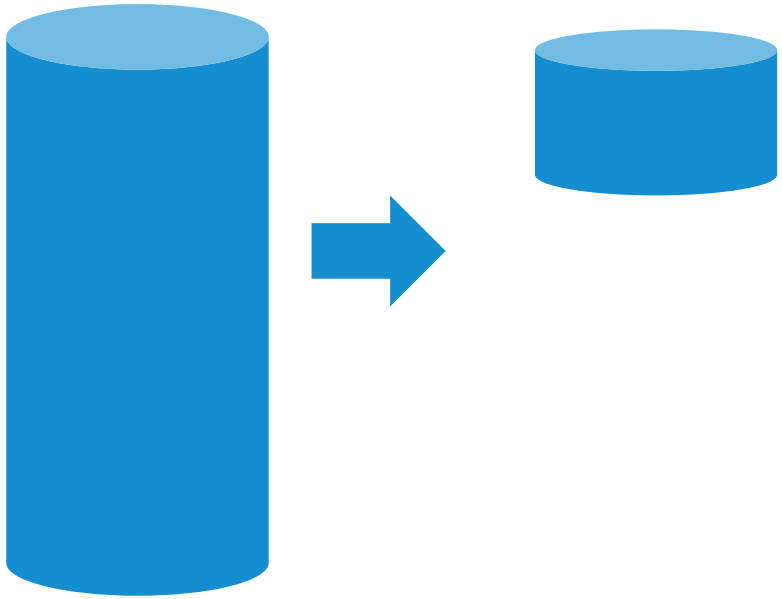
```
data C;
set A;
where ID in ("003","005");
run;

: There were 6 observations read from the data set WORK.A
  WHERE ID in ('003', '005');
: The data set WORK.C has 6 observations and 3 variables.
: DATA statement used (Total process time):
  real time           0.01 seconds
  cpu time            0.01 seconds
```

6 observations read

```
data C;
set A;
if ID in ("003","005");
run;

: There were 15 observations read from the data set WORK.A.
: The data set WORK.C has 6 observations and 3 variables.
: DATA statement used (Total process time):
  real time           0.00 seconds
  cpu time            0.00 seconds
```

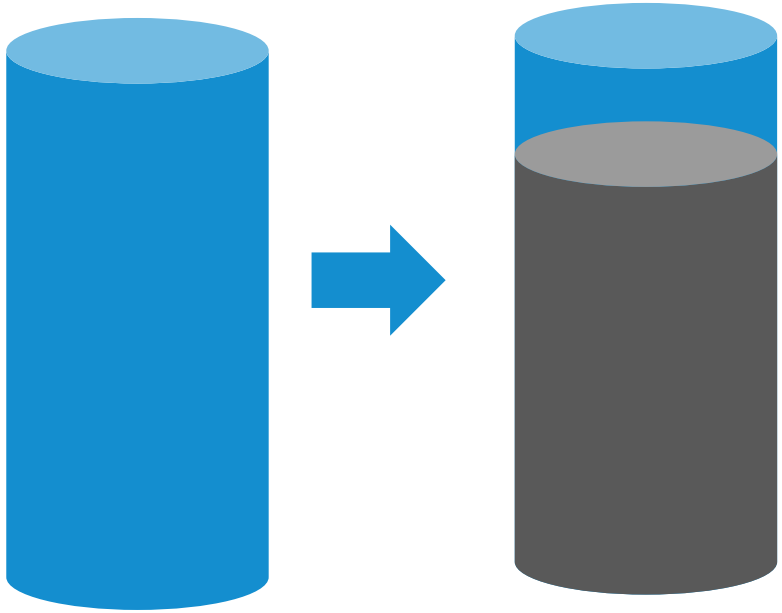
15 observations read

Where vs Subset if



Where

Where statement can reduce the amount of input



Subset if

Dataset: A

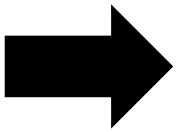
ID	TPT	VAL
002	1	63
002	2	89
002	3	84
001	1	71
001	2	44
001	3	80
003	1	77
003	2	24
003	3	46
005	1	17
005	2	90
005	3	50
004	1	47
004	2	45
004	3	32

Big

Dataset: B

ID
003
005

List data

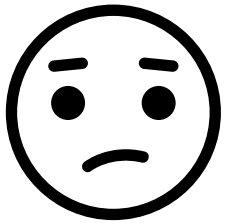


Dataset: C

ID	TPT	VAL
003	1	77
003	2	24
003	3	46
005	1	17
005	2	90
005	3	50

```
data C;
set A;
if 0 then set B;
if _N_=1 then do;
      declare hash h1(dataset:"B");
      h1.definekey("ID");
      h1.definedone();
end;
if h1.check()=0;
run;
```

I want to take advantage of where Statement..



Proc FCMP

```
proc fcmp outlib=work.functions.common;
function test(a,b);
 if a>b then c=a+b;
 else c=a*b;
 return(c);
endsub;
quit;

options cmplib = work.functions;
data C;
x=2;
y=3;
z=test(x,y);
output;
x=3;
y=2;
z=test(x,y);
output;
run;
```

FCMP can create user-defined functions and use them in data steps and proc SQL

x	y	z
2	3	6
3	2	5

```
proc fcmp outlib=work.functions.common;
function check_B(ID $);
 declare hash h1(dataset: "B");
 rc=h1.definekey("ID");
 rc=h1.definedone();
 if h1.check() eq 0 then return(1);
 else return(0);
endsub;
quit;


 data C;
 set A;
 where check_B(ID);
 run;

NOTE: There were 6 observations read from the data set WORK.A.
      WHERE CHECK_B(ID);
```

Wow!!

```
proc sql noprint;
 create table C as
 select *
 from A
 where check_B(ID);
quit;
```

Wow!!

SQL and HASH collaboration！ ！

```
proc sql;
 select NAME,AGE,lag(AGE)
 from sashelp.class;
quit;
```

```
156  proc sql;
157   select NAME,AGE,lag(AGE)
158   from sashelp.class;
ERROR: The LAG function is not supported in PROC SQL, it is only valid within the DATA step.
159  quit;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

```
proc sql;
 select NAME,AGE,lagn(AGE)
 from sashelp.class;
quit;
```

Name	Age	
Alfred	14	.
Alice	13	14
Barbara	13	13
Carol	14	13
Henry	14	14

Amazing!!

Dictionary Object

```
proc fcmp outlib=work.functions.common;
 function dcheck(var1,var2);
 declare dictionary d1;
  d1[1,63]=1;
  d1[3,32]=3;
  d1[2,45]=4;
  return(d1[var1,var2]);
 endsub;
quit;
```

Dictionary Image

key1	key2	data
1	63	1
3	32	3
2	45	4

```
data C;
set A;
where dcheck(TPT,VAL) >=2;
run;
```

ID	TPT	VAL
004	2	45
004	3	32

It seems that the used memory size is lighter than Hash

Analyze the SAS code

Test & Refactoring

Proc Scaproc

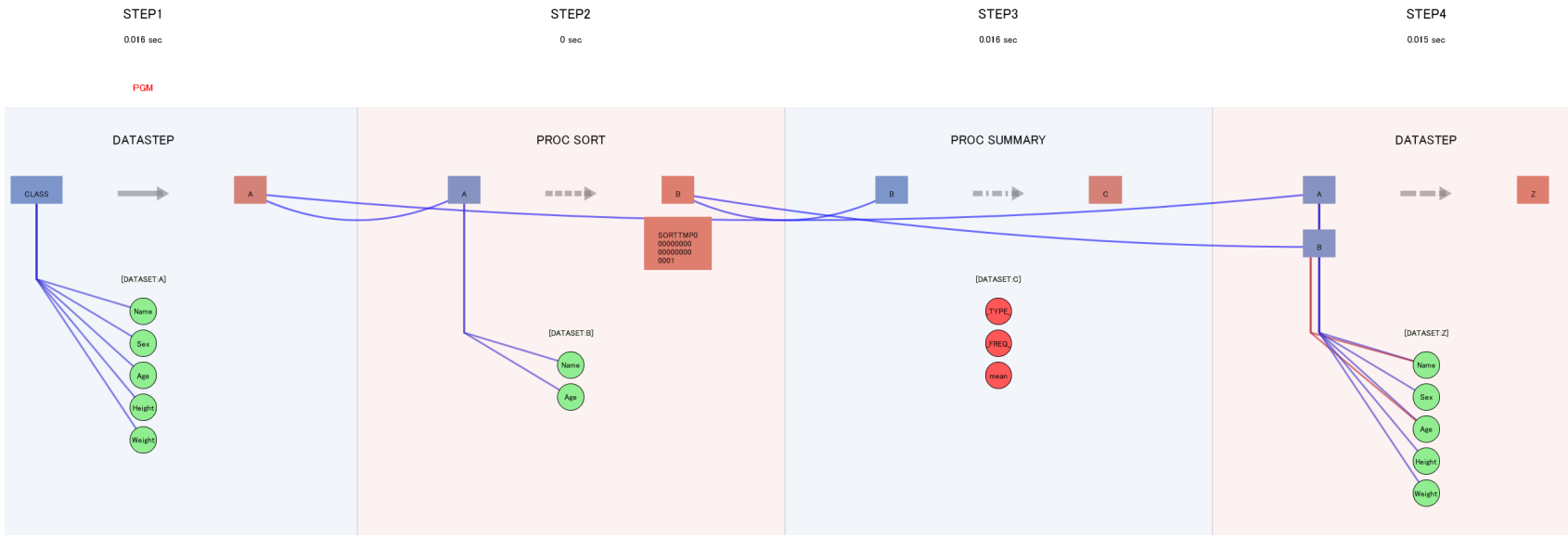
test.sas

```
data a;
set sashelp.class;
run;

proc sort data=a(keep=name age) out=b nodupkey;
  by age;
run;

proc summary data=b;
    var age;
    output out=c mean=mean;
  run;

 data z;
   merge a b;
   by name;
 run;
```

```
filename recfile "output.txt";

/*START*/
proc scaproc;
record  recfile
 attr
 opentimes
 intcon
 expandmacros
;
run;

%inc"test.sas";

/*END*/
proc scaproc;
 write;
run;
```

```
/* JOBSPLIT: JOBSTARTTIME 22MAR2018:18:39:18.05 */
/* JOBSPLIT: TASKSTARTTIME 22MAR2018:18:39:18.06 */
/* JOBSPLIT: FILE INPUT F:¥DS¥¥test.sas */
/* JOBSPLIT: DATASET INPUT SEQ #C00001.CLASS.DATA */
/* JOBSPLIT: LIBNAME #C00001 V9 'C:¥Program Files¥SASHome¥x86¥SASFoundation¥9.4¥nls¥ja¥sashelp' */
/* JOBSPLIT: CONCATMEM #C00001 SASHELP */
/* JOBSPLIT: LIBNAME SASHELP V9 '( 'C:¥Program Files¥SASHome¥x86¥SASFoundation¥9.4¥nls¥ja¥SASCFG' 'C:¥Program
/* JOBSPLIT: OPENTIME #C00001.CLASS.DATA DATE:22MAR2018:18:39:18.06 PHYS:C:¥Program Files¥SASHome¥x86¥SASFoun
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.A.DATA */
/* JOBSPLIT: LIBNAME WORK V9 'D:¥Users¥10089669¥AppData¥Local¥Temp¥224¥SAS Temporary Files¥_TD48368_EPS2X008_
/* JOBSPLIT: ATTR #C00001.CLASS.DATA INPUT VARIABLE:Name TYPE:CHARACTER LENGTH:12 LABEL:NAME FORMAT: INFORMAT
/* JOBSPLIT: ATTR #C00001.CLASS.DATA INPUT VARIABLE:Sex TYPE:CHARACTER LENGTH:4 LABEL:SEX FORMAT: INFORMAT: *
```

Data handling and SGPLOT



Time is up.

CONCLUSION

Handling Big Data is an artistic and creative work.

The above kanji is a motto of a professional Shogi (Japanese Chess) player, meaning “always try a new move”. I always wish to be like such a leading SAS programmer.

Thank you!

Yutaka Morioka

E-mail1: morioka038@eps.co.jp
E-mail2: sasyupi@gmail.com