SAS069-2024

Five Easy Ways to Make Your SAS Code Run Faster

Mark Jordan, SAS Institute, Inc.

Abstract

Don't you just hate waiting for your results after submitting a SAS program? I sure do! In this session, I'll demonstrate five easy "thumb rules" that you can easily use to help you write faster-running SAS code. These techniques are valid for code executed in SAS 9 or the SAS Viya Compute Server. I'll demonstrate head-to-head comparisons of the different techniques in SAS 9, using SAS datasets and Oracle tables as data sources. You'll see how simple coding choices affect execution time, and learn which choices provide the best performance in the circumstances we commonly encounter every day.

Decisions, decisions!

- KEEP statement or the KEEP= Dataset Option?
- Subsetting IF or a WHERE statement?
- WHERE expressions: SAS-specific or ANSI standard operators?
- DATA step or SQL query?
- What if my process is CPU-bound?

In this presentation, I'll compare the impact on processing speed of several techniques we choose from every day. We'll try it using both SAS datasets and database data so we can get a good idea of what produces the fastest result under most frequently encountered circumstances. My hope is that together we can agree on 5 "thumb rules" we can use to ensure our code run fast and efficiently when we don't have the time for formal benchmarking.

The Demo Code

```
/*********************************************************
 System Setup
*********************************************************/
%let path=/home/student/Demos/faster-code;
cas mySession ;

libname db oracle path="client.demo.sas.com:1521/orcl" user=student
pw=Metadata0;
libname sas "&path/data";
caslib mycas path="&path/data" libref=mycas;
options sastrace=',,,d' sastraceloc=saslog nostsuffix;

/*********************************************************
 1. KEEP vs. (KEEP=)
*********************************************************/
/* With SAS data */
/* real time ~1.5 seconds */
data _null_;
   set sas.wide;
   keep id;
run;

/* real time ~1.12 seconds */
data _null_;
   set sas.wide(keep=ID);
run;

/* With DBMS data */
/* real time ~10 seconds! */
data _null_;
   set db.wide;
   keep id;
run;
```

```
/* real time ~0.1 second! */
data _null_;
   set db.wide(keep=ID);
run;

/* Thumb rule: KEEP= on SET and KEEP statement yield the same result, use
KEEP= */

/*********************************************************
 2. IF vs. WHERE
*********************************************************/
/* With SAS data */
/* real time ~0.3 second */
data _null_;
   set sas.narrow;
   if id=4;
run;

/* real time ~0.2 second */
data _null_;
   set sas.narrow;
   where id=4;
run;

/* With DBMS data */
/*  real time ~3.6 second */
data _null_;
   set db.narrow;
   if id=4;
run;

/*  real time ~0.1 second */
data _null_;
   set db.narrow;
   where id=4;
run;

/* Thumb rule: If subsetting IF and WHERE yield the same result, use WHERE */

/*********************************************************
 3. SAS vs. ANSI WHERE expressions for database data
*********************************************************/
/* DATA step */
/* real time ~0.1 second */
data _null_;
   file print;
   set db.big_customer_dim (keep=customer_id customer_name);
   where scan(customer_name,1) ='Silvestro';
   put customer_id customer_name;
run;

/* real time ~0.02 second */
data _null_;
   file print;
   set db.big_customer_dim (keep=customer_id customer_name);
   where customer_name like 'Silvestro %' ;
   put customer_id customer_name;
run;
```

```
/* SQL */
/* real time ~0.1 second */
proc sql;
select customer_name
   from db.big_customer_dim
   where scan(customer_name,1) ='Silvestro'
;
quit;

/* real time ~0.02 second */
proc sql;
select customer_name
   from db.big_customer_dim
   where customer_name like 'Silvestro %'
;
quit;

/* Thumb rule: Keep WHERE exression ANSI if possible */

/*************************************************************
 4. DATA step vs. SQL
*************************************************************/
/* My DATA step code is pretty fast */
/* real time ~0.35 second */
data summary;
   keep Count Total Average;
   set sas.narrow end=last;
   Total+ru;
   if not missing (ru) then Count+1;
   if last then do;
      average=Total/Count;
      output;
   end;
run;

/* This PROC SQL query is slower */
/* real time ~0.5 second */
proc sql;
create table summary_sql as
select count(ru) as Count
      ,sum(ru) as Total
      ,mean(ru) as average
   from sas.narrow
;
quit;

/* But if the data source is a DBMS, there is always a clear winner: */
/*  real time ~4 seconds */
data summary;
   keep Count Total Average;
   set db.narrow end=last;
   Total+ru;
   if not missing (ru) then Count+1;
   if last then do;
      average=Total/Count;
      output;
   end;
run;
```

```
/* This PROC SQL query goes straight in-database */
/* real time ~0.2 second */
proc sql;
create table summary_sql as
select count(ru) as Count
      ,sum(ru) as Total
      ,mean(ru) as average
   from db.narrow
;
quit;

/* Thumb rule: If DATA step and SQL produce the same single result set, use
SQL */

/* real time 4.8 seconds */
data group1 group2 group3 group4 group5 group6 group7 group8 group9 group10;
   set db.narrow;
   select;
      when (ru<=1) output group1;
      when (ru<=2) output group2;
      when (ru<=3) output group3;
      when (ru<=4) output group4;
      when (ru<=5) output group5;
      when (ru<=6) output group6;
      when (ru<=7) output group7;
      when (ru<=8) output group8;
      when (ru<=9) output group9;
      when (ru<=10) output group10;
      otherwise;
   end;
run;

/* real time 5.8 seconds */
proc sql;
create table group1 as
   select * from db.narrow where ru<=1;
create table group2 as
   select * from db.narrow where ru>1 and ru <=2;
create table group3 as
   select * from db.narrow where ru>2 and ru <=3;
create table group4 as
   select * from db.narrow where ru>3 and ru <=4;
create table group5 as
   select * from db.narrow where ru>4 and ru <=5;
create table group6 as
   select * from db.narrow where ru>5 and ru <=6;
create table group7 as
   select * from db.narrow where ru>6 and ru <=7;
create table group8 as
   select * from db.narrow where ru>7 and ru <=8;
create table group9 as
   select * from db.narrow where ru>8 and ru <=9;
create table group10 as
   select * from db.narrow where ru>9 and ru <=10;
quit;

/* Thumb rule: If multiple result sets are required, use DATA step. */
```

```
/*********************************************************
 5. CPU-bound processes
*********************************************************/
/* This scoring DATA step is CPU-bound (real time ~= CPU time). */
/* real time  ~38 seconds */
/* cpu time   ~43 seconds */
data t1;
   array score[0:100];
   set sas.narrow END=LAST;
   do i=LBOUND(SCORE) to hbound(score);
      Score[i]= (SQRT(((id * ru * rn) / (id + rn + ru))*ID))*(SQRT(((id * ru
* rn) / (id + rn + ru))*ID));
   end;
   count+1;
   if last then put 'Data step processed ' count 'observations.';
   drop i count;
run;

/* Parallel processing with DS2 */

/* Create thread  */
/* real time ~0.1 second */
proc ds2;
thread MyThreadProgram/overwrite=yes;
   dcl bigint count;
   drop count;
   vararray double score[0:100] score0-score100;
   method run();
      dcl int i;
      set sas.narrow;
         do i=LBOUND(SCORE) to hbound(score);
            Score[i]= (SQRT(((id * ru * rn) / (id + rn + ru))*ID))
                      *(SQRT(((id * ru * rn) / (id + rn + ru))*ID));
         end;
         count+1;
   end;
   method term();
      put 'Thread' _threadid_ ' processed' count 'observations.';
   end;
endthread;
run;
quit;

/* Scoring in threads */
/* real time ~8 seconds */
proc ds2;
/*Multi-threaded*/
data t2/overwrite=yes;
   dcl thread MyThreadProgram th;
   method run();
      set from th threads=&sysncpu;
   end;
enddata;
run;
quit;
```

Thumb Rules for Faster Processing

- If the KEEP statement and KEEP= dataset option on the input dataset produce the same results, use KEEP=
- If the subsetting IF and WHERE statements produce the same results, use WHERE.
- When construction a WHERE expression, use ANSI standard operators whenever possible.
- DATA step or SQL Query?
  - If producing a single result set from DBMS data, use SQL.
  - If producing multiple result sets, use a single DATA step.
- For CPU-bound processes:
  - In SAS® Viya®, process in CAS
  - In SAS 9, consider parallel processing with DS2

5 Easy Way to Make Your SAS Code Run

Faster

Email: Mark.Jordan@sas.com
X.com: @SASJedi
Blog: http://sasjedi.blog
Author Info: http://support.sas.com/jordan

Get the ZIP file with the PDF and code!
Scan the QR code or got to
https://bit.ly/SASJediFiveFastTips

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

Mark Jordan
SAS Institute, Inc.
Email: mark.jordan@sas.com
X: @SASJedi