TOP 10 WAYS TO OPTIMIZE YOUR SAS CODE

RON COLEMAN
SR. SOLUTIONS ARCHITECT

CAN I MAKE SAS EFFICIENT?

- **YES!**
  - We control what resources are used
  - We need to understand how SAS works
  - We need to understand how to better use resources

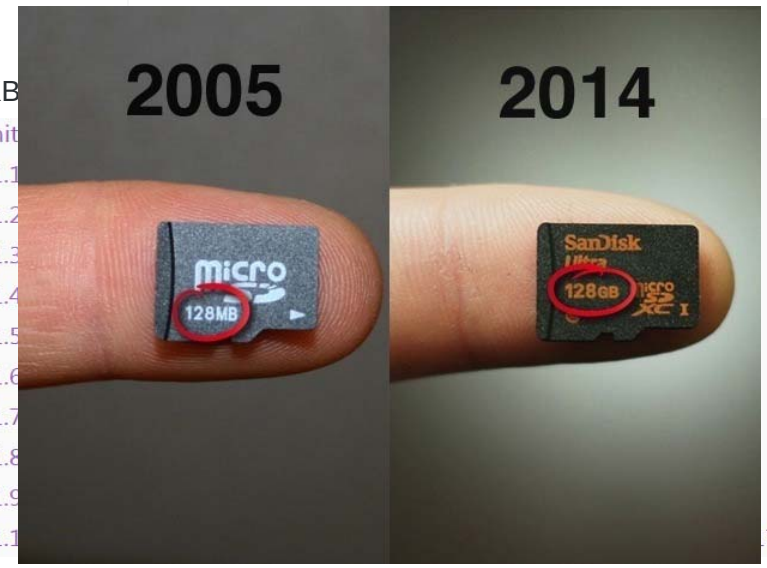
MEASURABLE RESOURCES

- Timings
  - CPU
    - » Time processor is actually working
  - Real or Wall
    - » Actual time (clock on the wall)

MEASURABLE RESOURCES

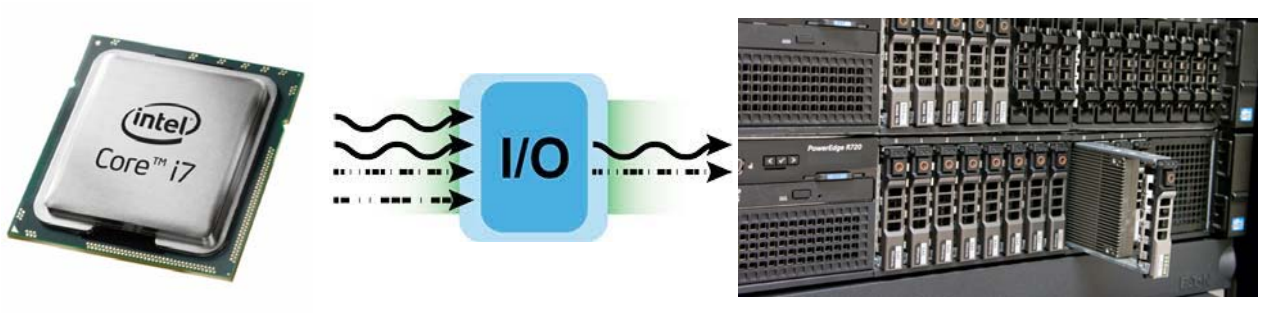
- Disk
  - Space (KB

1 Unit
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
1.1 ... 76 bytes)



MEASURABLE RESOURCES

- I/O
  - Passing data between CPU and Disk



EXAMPLE OF I/O VS CPU

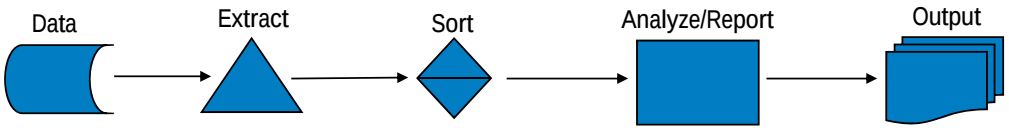
- Single processor laptop (2.8GHz)
- Sample program has 100 long character strings
- Writes 100,000 rows to SAS table

```
NOTE: The data set WORK.SIZETEST has 100000 observations and 101 variables.
NOTE: DATA statement used:
      real time          1:43.95
      cpu time            5.99 seconds
```

SAS IS I/O INTENSIVE

```
data temp1;
   infile '/data/demodata';
   input value;
run;
proc sort data=temp1;
   by value;
run;
proc means data=temp1;
   by value;
run;
```

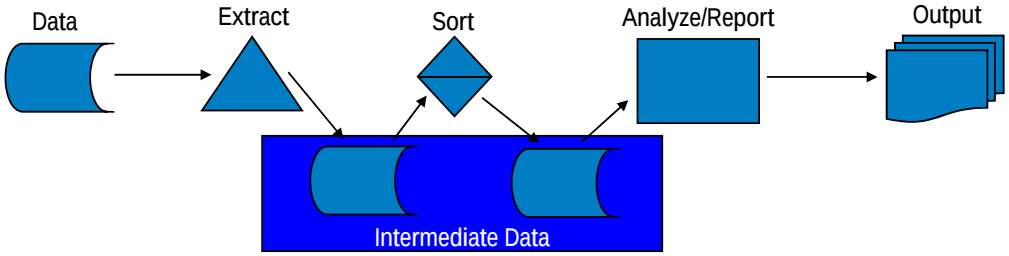
- What we see:



SAS IS I/O INTENSIVE

```
data temp1;
   infile '/data/demodata';
   input value;
run;
proc sort data=temp1;
   by value;
run;
proc means data=temp1;
   by value;
run;
```

- What is really happening!

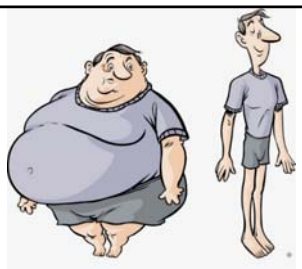


KEYS TO REDUCING I/O

- Amount of data we use
- Number of times we touch the data
- New data we create

#1 – USE ONLY DATA THAT IS NEEDED

#1 – USE ONLY DATA THAT IS NEEDED



- Limiting the columns
  - Flat Files
    - » Use @ pointer operator to skip unnecessary fields

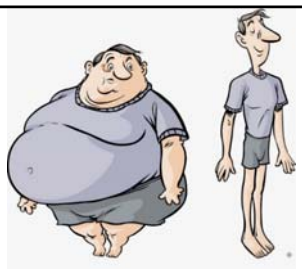
```
filename datain 'F:\LargeData\DetailData2011.txt' lrecl=2048;
data work.detail;
   infile datain;
   input @57 date YYMMDD10. @123 cust_id $15. @897 balance 16.2;
run;
```


#1 – USE ONLY DATA THAT IS NEEDED

- Limiting the columns
  - Delimited
    - » Use dummy variables of length 1 to reduce size

```
filename csvin 'F:\LargeData\SummaryInfo.csv';
data work.SummaryInfo(keep=date cust_id balance);
   infile csvin dsd;
   input date :YYMMDD10. dummy :$1. cust_id :$15.
      (dummy dummy dummy) (:$1.) balance :16.2;
run;
```

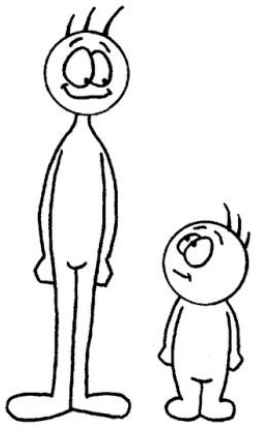
#1 – USE ONLY DATA THAT IS NEEDED



- Limiting the columns
  - SAS Tables
    - » Use KEEP= or DROP= dsoptions on the Input and Output tables

```
libname sasdata '/opt/sasdata';
data work.subset;
   set sasdata.ManyColumns(KEEP=date cust_id balance);
   *. . . ;
run;
```


#1 – USE ONLY DATA THAT IS NEEDED



- Limiting the rows
  - Flat Files
    - » Read key fields first
    - » Make decision
    - » Read remaining fields

```
data thisyear;
   infile 'All_Sales';
   input @27 year 4. @;
   if year = 2012;
   input …;
   …
run;
```

#1 – USE ONLY DATA THAT IS NEEDED

- Limiting the rows
  - SAS Tables
    - Use a WHERE statement/dsoption
      - Columns must exist on table
      - May contain functions
      - Restricts what is sent into PDV
    - Use a subsetting IF when WHERE is not possible
      - Executes after row is read from table

```
data work.subset;
   set sasdata.ManyColumns(KEEP=date cust_id balance);
   where date between '1JAN2010'd and '31dec2010'd;
run;
```


COMPRESS=YES

- Happens in memory
  - Uses CPU – adds approximately 1-3%
  - Reduces I/O
- Best for tables with many character columns
- Can increase table size for numeric rich tables
  - Improved processing in SAS 9.2 – will forego compression
- COMPRESS=BINARY
  - Requires more processing
  - Great for numeric rich tables

#2 – PROCESS ONCE

- Do as much processing as possible in each step

Wrong Way!

```
data one;
   set original;
   revenue = price - cogs;
run;

data two;
   set one;
   margin = revenue / price;
run;
```

Right Way!!!

```
data one;
   set original;
   revenue = price - cogs;
   margin = revenue / price;
run;
```


#2 – PROCESS ONCE

```
data youngfemales;
   set one;
   if  gender='F' and age <=18;
run;

data youngmales;
   set one;
   if  gender='M' and age <=18;
run;

data oldfemales;
   set one;
   if  gender='F' and age >18;
run;

data oldmales;
   set one;
   if  gender='M' and age >18;
run;
```

#2 – PROCESS ONCE

```
data youngfemales youngmales oldfemales oldmales;
   set one;

   if  gender='F' and age <= 18 then
      output youngfemales;
   else if  gender='F' and age > 18 then
      output oldfemales;
   else if  gender='M' and age <= 18 then
      output youngmales;
   else if  gender='M' and age > 18 then
      output oldmales;
run;
```


#2 – PROCESS ONCE

- Using Pass-Thru SQL

```
proc sql;
   connect to oracle (authdomain="oraauth" PATH=ESR);

   create table Work.Leads_2012 as
   select *
   from connection to oracle (
      . . .
   );
quit;

data Work.Leads_2012;
   set Work.Leads_2012;
   startdate = datepart(startdate);
   enddate   = datepart(enddate);
   format startdate enddate date9.;
run;
```

#2 – PROCESS ONCE

- Use the SAS part of the query!

```
/* Do this in one pass of the data! */
proc sql;
   connect to oracle (authdomain="oraauth" PATH=ESR);

   create table Work.Leads_2012 as
   select . . .
         , datepart(startdate) as startdate format=date9.
         , datepart(enddate)   as enddate   format=date9.
   from connection to oracle (
      . . .
   );
quit;
```


#2 – PROCESS ONCE

- Or use a view!

```
proc sql;
   connect to oracle (authdomain="oraauth" PATH=ESR);

   create view Work.Leads_2012_V as
   select *
   from connection to oracle (
      . . .
   );
quit;

data Work.Leads_2012;
   set Work.Leads_2012_V;
   startdate = datepart(startdate);
   enddate   = datepart(enddate);
   format startdate enddate date9.;
run;
```

#3 – USE THE CORRECT TOOLS

- NOT a DATA step
  - Must process each row in the table
  - Creates another copy of the table

```
data sasdata.ManyColumns;
   set sasdata.ManyColumns;
   format date weekdate18.
          balance dollar 12.2;
   label cust_id = 'Customer ID';
run;
```

- Use DATASETS to assign column attributes
  - Formats
  - Labels

```
proc datasets lib=sasdata nolist;
   modify ManyColumns;
      format date weekdate18.
             balance dollar 12.2;
      label cust_id = 'Customer ID';
quit;
```


#4 – NO EXTRA BAGGAGE

- Long descriptive values
  - Better suited for formats
  - Add length to every row in the table
- Keep only what is needed
  - Drop index variables
- Variable lengths
  - Determine if numeric or character is better
    - » Conversion costs
  - Long character values often have unused space

#5 – SORT ONLY WHEN REQUIRED

- Flat File
  - If data is already ordered correctly use the SORTEDBY= dsoption

```
filename datain 'F:\LargeData\DetailData2011.txt' lrecl=2048;
data work.detail(sortedby=cust_id);
   infile datain;
   input @57 date YYMMDD10. @123 cust_id $15. @897 balance 16.2;
run;
```

#5 – SORT ONLY WHEN REQUIRED

- SAS Tables
  - Consider using indexes if many sort orders are required
- DBMS Tables (DB2, Oracle, SQL Server, etc.)
  - Use an ORDER BY clause when extracting data and SAS will recognize the data as sorted

#6 – USE FORMATS FOR GROUPING IN REPORTS

- Recode values

  'EBI' = 'SAS Enterprise BI Server'

- Grouping character values

```
value $region
    'ME','NH','VT','MA','RI','CT',
    'NY','NJ','PA'                    = 'Northeast Region'
    'DE','MD','VA','WV','NC','SC',
    'GA','FL','TN','AL','MS','LA'     = 'Southeast Region'
    'MI','OH','KY','IN','WI','IL',
    'MN','IA','MO','ND','SD','NE',
    'KS'                              = 'Midwest Region'
    'AR','OK','TX'                    = 'Midsouth Region'
    'MT','WY','CO','UT','NM'          = 'Mountain Region'
    'CA','OR','WA','ID','NV','AZ'     = 'West Region'
 ;
```


#6 – USE FORMATS FOR GROUPING IN REPORTS

```
proc freq data=sashelp.zipcode;
   table statecode;
   format statecode $region.;
```

The FREQ Procedure

Two-letter abbrev. for state name.

STATECODE	Frequency	Percent	Cumulative Frequency	Cumulative Percent
AK	272	0.66	272	0.66
Southeast Region	9692	23.37	9964	24.03
Mid-south Region	4094	9.87	14058	33.90
West Region	4945	11.93	19003	45.83
Mountain Region	2025	4.88	21028	50.71
Northeast Region	7392	17.83	28420	68.54
DC	278	0.67	28698	69.21
FM	4	0.01	28702	69.22

#6 – USE FORMATS FOR GROUPING IN REPORTS

- Grouping numeric values

```
value agerange
     Low - <18      = 'Minor'
     18  - <30      = 'Young Adult'
     30  - <45      = 'Early Family'
     45  - <60      = 'Middle Age'
     60  - high     = 'Just getting started!'
  ;
```


#7 – USE SAS DATE VALUES

- Valid from 1582 A.D. to 19900 A.D.
- Many formats available for grouping/reporting

```
Date.       11JUL18
Date9.      11JUL2018
Worddate.       July 11, 2018
Weekdate.       Wednesday, July 11, 2018
Month.       7
Monname.        July
Monname3.   Jul
monyy7.     JUL2018
```

#7 – USE SAS DATE VALUES

```
proc tabulate data=sashelp.citiwk;
   class date;
   var mf3505;
   tables date, mf3505*(min mean max);
   format date monyy7.;
run;
```

	MONEY STOCK:M1(CURRENCY+DEMAND DEP+OTHER		
	Min	Mean	Max
Date of Observation	620.80	620.80	620.80
DEC1985			
JAN1986	620.50	621.05	622.20
FEB1986	622.80	625.90	628.30
MAR1986	631.50	635.56	639.50
APR1986	639.40	643.00	648.10
MAY1986	651.90	654.17	656.00

#7 – USE SAS DATE VALUES

```
proc tabulate data=sashelp.citiwk;
   class date;
   var mf3505;
   tables date, mf3505*(min mean max);
   format date year4.;
run;
```

	MONEY STOCK:M1(CURRENCY+DEMAND DEP+OTHER		
	Min	Mean	Max
Date of Observation	620.80	620.80	620.80
1985			
1986	620.50	668.75	739.20
1987	726.40	744.42	765.80
1988	753.60	776.16	787.40
1989	773.00	783.59	796.90
1990	793.80	812.89	827.80
1991	822.00	861.40	901.30
1992	903.50	908.15	912.80

#8 – REDUCE UNNECESSARY WORK

- Avoid things like:

```
Name = left(trim(name));
```

```
$CHAR. Informat (Only needed in rare cases)
```

- Learn what functions/tools are available

Scan()	Index()	Substr()
Verify()	Translate()	Intnx()
Upcase()	Exist()	Compress()
Notname()	Propcase()	Anyalpha()

New in SAS 9!

#9 – LEARN MORE FEATURES

- DBMS access
  - Use GROUP BY to allow the DBMS to presummarize the data
  - Use ORDER BY to presort the data
- SAS
  - Create multiple levels of summary with PROC SUMMARY
  - Create multiple reports PROC FREQ and TABULATE

#10 – MANAGE DISK SPACE

- Delete old data along the way
- Recreate tables with the same name

```
Data one;
   set one;
   /* processing goes here */
Run;
```

- Only keep what you need!

SAS System and Data Set Options

VIEWING OPTION SETTINGS

proc options group=performance;

run;

SAS (r) Proprietary Software Release 9.2 TS2M3

ARMAGENT= ARM Agent to use to collect ARM records
ARMLOC=ARMLOG.LOG Identify location where ARM records are to be written
ARMSUBSYS=(ARM_NONE)
Enable/Disable ARMing of SAS subsystems
BUFNO=1 Number of buffers for each SAS data set
BUFSIZE=0 Size of buffer for page of SAS data set
CGOPTIMIZE=3 Control code generation optimization
CMPMODEL=BOTH Identify CMP model storage type
CMPOPT=(NOEXTRAMATH NOMISSCHECK NOPRECISE NOGUARDCHECK NOFUNCDIFFERENCING)
Enable SAS compiler performance optimizations
COMPRESS=NO Specifies whether to compress observations in output SAS data sets
CPUCOUNT=8 Number of processors available.
NODBIDIRECTEXEC Do not use SQL optimization with SAS/Access engine
DBSLICEPARM=(THREADED_APPS, 2)
Alter DBMS engine threaded read behavior by expanding or disallowing
threaded reads.
MAXSEGRATIO=75 SPDE pre-evaluation phase time ratio
MEXECSIZE=65536 Maximum size for a macro to execute in memory
MINPARTSIZE=16777216
Minimum partition size when creating SPDE files
SORTSIZE=67108864 Size parameter for sort
SPDEINDEXSORTSIZE=33554432
Identifies memory to be used for SPDE asynchronous index create or append
SPDEMAXTHREADS=0 Maximum number of threads for SPDE processing
SPDESORTSIZE=33554432
Memory for SPDE sort operations
SPDEUTILLOC= Location where SPDE temporary utility files are created

VIEWING OPTION SETTINGS

proc options group=performance;

run;

SAS (r) Proprietary Software Release 9.2 TS2M3

BUFNO=1 Number of buffers for each SAS data set

BUFSIZE=0 Size of buffer for page of SAS data set

COMPRESS=NO Specifies whether to compress observations in output SAS data sets

MEMSIZE=536870912 Specifies the limit on the total amount of memory to be used by the SAS System

COMPRESS=YES

- Happens in memory
  - Uses CPU – adds approximately 1-3%
  - Reduces I/O
- Best for tables with many character columns
- Can increase table size for numeric rich tables
  - Improved processing in SAS 9.2 – will forego compression
- COMPRESS=BINARY
  - Requires more processing
  - Great for numeric rich tables

BUFSIZE AND BUFNO

- BUFNO=
  - Specifies the number of buffers when reading/writing a table
- BUFSIZE=
  - Specifies the buffer size when writing a table
  - Default is recommended to be BUFSIZE=64K
- Should only be used as Data Set Options (dsoption)
  - For very large (100GB+) tables
  - Experiment to find best setting (256K – 1M)
  - Must be set when creating the table

```
Data permdata.Very_Large_Table(BUFNO=10 BUFSIZE=512K);
```

MEMORY OPTIONS

proc options group=memory;

run;

SAS (r) Proprietary Software Release 9.2 TS2M3

SORTSIZE=67108864 Size parameter for sort

SUMSIZE=0 Upper limit for data-dependent memory usage during summarization

MAXMEMQUERY=0 Maximum amount of memory returned when inquiring as to available space

MEMBLKSZ=16777216 Size of memory blocks allocated to support MEMLIB and MEMCACHE options.

MEMMAXSZ=2147483648

Maximum amount of memory allocated to support MEMLIB and MEMCACHE options.

LOADMEMSIZE=0 Suggested memory limit for loaded SAS executables

MEMSIZE=536870912

SURVEY

http://www.surveygizmo.com/s3/1202374/AmeriHealth-3-25-13-Top-10-Ways-to-Optomize-Your-Code