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, MB, GB, TB, ?)
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

• What we see:

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

SAS IS I/O INTENSIVE

• What is really happening!

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

```sas
c 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;
```

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

```sas
c 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 (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

```sas
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

```sas
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

```sas
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!

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

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

Right Way!!!

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

```sas
#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!

```sas
/* 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;
```

- Or use a view!

```sas
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

```sas
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

```sas
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

```sas
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;
```

- 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

```sas
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';
```

---

```sas
proc freq data=sashelp.zipcode;
  table statecode;
  format statecode $region.;
run;
```
#6 – USE FORMATS FOR GROUPING IN REPORTS

- Grouping numeric values

```r
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

<table>
<thead>
<tr>
<th>Format</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>11JUL18</td>
</tr>
<tr>
<td>Date9</td>
<td>11JUL2018</td>
</tr>
<tr>
<td>Worddate</td>
<td>July 11, 2018</td>
</tr>
<tr>
<td>Weekdate</td>
<td>Wednesday, July 11, 2018</td>
</tr>
<tr>
<td>Month</td>
<td>7</td>
</tr>
<tr>
<td>Monname</td>
<td>July</td>
</tr>
<tr>
<td>Monname3</td>
<td>Jul</td>
</tr>
<tr>
<td>monyy7</td>
<td>JUL2018</td>
</tr>
</tbody>
</table>
#7 – USE SAS DATE VALUES

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

<table>
<thead>
<tr>
<th>Date of Observation</th>
<th>Min</th>
<th>Mean</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC1985</td>
<td>620.80</td>
<td>620.80</td>
<td>620.80</td>
</tr>
<tr>
<td>JAN1986</td>
<td>620.50</td>
<td>621.05</td>
<td>622.20</td>
</tr>
<tr>
<td>FEB1986</td>
<td>622.80</td>
<td>625.90</td>
<td>628.30</td>
</tr>
<tr>
<td>MAR1986</td>
<td>631.50</td>
<td>635.56</td>
<td>639.50</td>
</tr>
<tr>
<td>APR1986</td>
<td>639.40</td>
<td>643.00</td>
<td>648.10</td>
</tr>
<tr>
<td>MAY1986</td>
<td>651.90</td>
<td>654.17</td>
<td>656.00</td>
</tr>
</tbody>
</table>

#7 – USE SAS DATE VALUES

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

<table>
<thead>
<tr>
<th>Date of Observation</th>
<th>Min</th>
<th>Mean</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1985</td>
<td>620.80</td>
<td>620.80</td>
<td>620.80</td>
</tr>
<tr>
<td>1986</td>
<td>620.50</td>
<td>668.75</td>
<td>739.20</td>
</tr>
<tr>
<td>1987</td>
<td>726.40</td>
<td>744.42</td>
<td>765.80</td>
</tr>
<tr>
<td>1988</td>
<td>753.60</td>
<td>776.16</td>
<td>787.40</td>
</tr>
<tr>
<td>1989</td>
<td>773.00</td>
<td>783.59</td>
<td>796.90</td>
</tr>
<tr>
<td>1990</td>
<td>793.80</td>
<td>812.89</td>
<td>827.80</td>
</tr>
<tr>
<td>1991</td>
<td>822.00</td>
<td>861.40</td>
<td>901.30</td>
</tr>
<tr>
<td>1992</td>
<td>903.50</td>
<td>908.15</td>
<td>912.80</td>
</tr>
</tbody>
</table>
#8 – REDUCE UNNECESSARY WORK

- Avoid things like:
  
  ```
  Name = left(trim(name));
  ```

  $CHAR. Informat (Only needed in rare cases)

- Learn what functions/tools are available

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan()</td>
<td>Index()</td>
<td>Substr()</td>
</tr>
<tr>
<td>Verify()</td>
<td>Translate()</td>
<td>Intnx()</td>
</tr>
<tr>
<td>Upcase()</td>
<td>Exist()</td>
<td>Compress()</td>
</tr>
<tr>
<td>Notname()</td>
<td>Propcase()</td>
<td>Anyalpha()</td>
</tr>
</tbody>
</table>

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

- Only keep what you need!
**VIEWING OPTION SETTINGS**

```sas
proc options group=performance;
run;
```

**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 permd ata.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

TOP 10 WAYS TO OPTIMIZE YOUR SAS CODE