DATA MANAGEMENT • BIOSTATISTICS • PROGRAMMING • MEDICAL WRITING • PHARMACOVIGILANCE

Quanticate

A PASSION FOR EXCELLENCE

How to Deal with Large Datasets

Gloria D' Souza

Global Solutions from the World's Largest Data-Dedicated CRO

www.quanticate.com

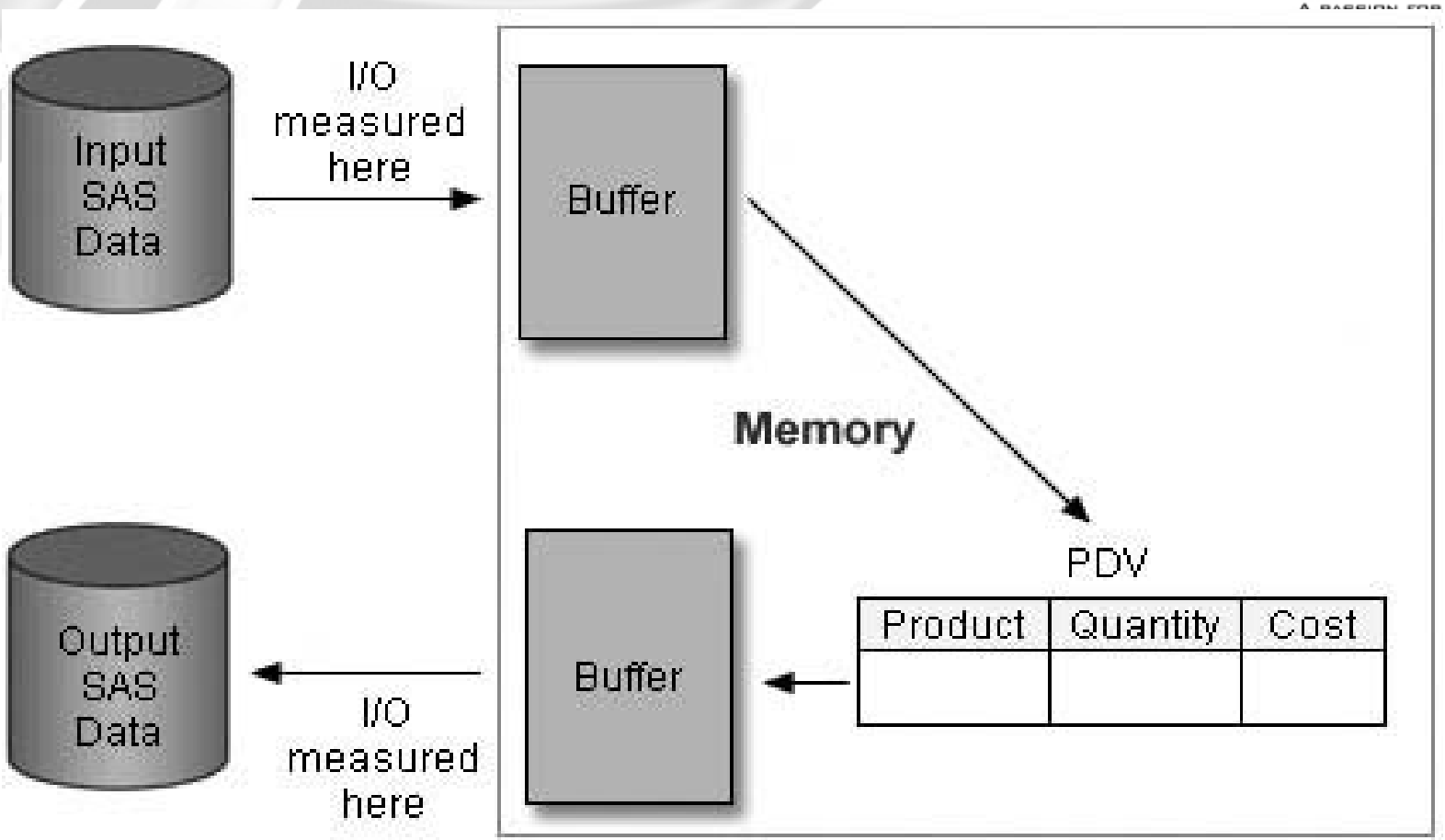
What we will cover

- Basic Concepts
- How SAS works
- What happens when data is large
- Ways to deal with Large datasets
  - Decreasing Data Storage Space
  - Decreasing I/O and CPU
  - Decreasing Memory Usage (out of scope of this presentation)
  - Tips and Tricks
- Understanding Trade-Offs
- Summary

Basic Concepts

- **Program Data Vector (PDV)**
- **Page**
- **CPU time**
- **Real Time**
- **MEMORY**
- **Data Storage**
- **I/O**
- **Buffer**

How SAS works



What happens when data is large?

Quanticate
A PASSION FOR EXCELLENCE

- More data storage required
- Multiple pages need to be read into input bu… I/C…e
- More observations are processed in … -> CPU … e i…eases
- More data is stored in me…ry … Bigge… ff…rs are required-> more memory utilised
- More real ti…e

Thus … exe…tion … and slowing of resources

REDUCE THIS

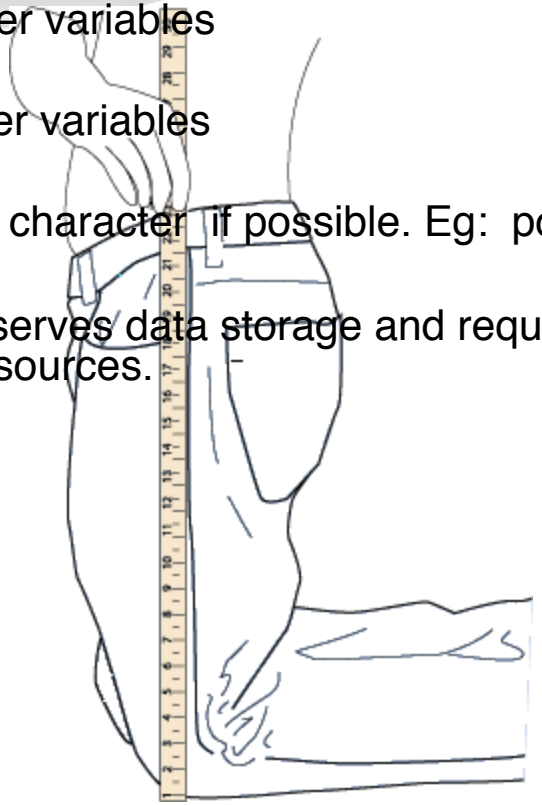
Decreasing Data Storage Space

- Length Statement
- Compress
- SAS Views
- Other Techniques
  - Normalisation
  - Using SAS functions efficiently
  - Proc SQL versus Data step

Length Statement

Quanticate
A PASSION FOR EXCELLENCE

- Optimal Length for character variables
- Reduce the length of integer variables
- Store numeric variables as character if possible. Eg: population flags
- Note: While doing this conserves data storage and requires less I/O to read, it requires additional CPU resources.



Dataset Compression

Quanticate
A PASSION FOR EXCELLENCE

- Compression in SAS works by finding repeated values and replacing that value with number that are same and the initial value

So: Apple, Apple, Apple
Becomes: 3 Apples.
However: Apple, Pear, Apple, Pear, Apple
Becomes: 1 Apple, 1 Pear, 1 Apple, 1 Pear, 1 Apple.

- When compression is used the SAS System does not 'honour' distinctions between each variable in an observation.
- Need to ensure strategic placing of variables
- If used incorrectly can increase the size
- Compressing datasets must be uncompressed before they can be used.
- Compressing datasets saves I/O and disk space but uses more CPU

Types of compression

YES I CHAR

- uses RLE (Run Length Encoding).

Thus : WWWBBHHHMMSS [12bytes]
becomes:
3W2B3H2M2S [10bytes]

- Use this compression algorithm for character data.

- Example:

```
data work.temp (compress=yes);
set work.temp;
run;
```

Types of compression

BINARY

- combines RLE and sliding-window compression to compress the file.
- For example: @can count countville count the countably infinite count " repeats the string ”count" five times.
- RLE would not be able to compress this since it only searches for characters within a word. But Sliding-window compression will represent the phrase ‘count’ with a small code.
- **Tip:** This method is highly effective for numeric data.

Eg: data work.temp (compress=binary);
set work.temp;
run;

Differences Observed when using CHAR and BINARY

Original Size	Number of Variables	Number of Observations	Using Compress=CHAR	Using Compress=BINARY
682 MB	23 variables all char	1397106	318MB (Reduced 54%)	337MB (Reduced 51%)
150MB	10 numeric and 2 char	1397106	106MB (Reduced 30%)	97MB (Reduced 36%)

Compression Trade-offs

UNCOMPRESSED	COMPRESSED
More disk storage	Less disk space
Deleted space is not reused	Deleted space can be reused.
Updated observation fits in original location	Updated observation may be moved to a new location. So sometimes more I/O
Less CPU time	More CPU time (because a data set must be uncompressed to be processed)
New observations are always inserted at the end of the file	New observations might not be inserted at the end of the file if we use REUSE=YES option

SAS Views

Quanticate
A PASSION FOR EXCELLENCE

- SAS Views are created in exactly the same way as datasets but they contain no actual data, only the code that is required to create the data.
- So each time you access a view, the code that created it is re-run and the data refreshed.
- SAS views can be used to great effect when accessing external files.
- Example:

```
data work.temp / view=work.temp;
infile 'inputdata.txt' delimiter=',' ;
input pt dob weight;
run;
```

- Syntax to see how the view was created

```
proc sql;
describe view sasuser.faview;
```

Other Techniques

Data Normalisation

- Store big length variables with an identifier in one dataset and merge it with main dataset only when required.

- Ideally used in Labs data

PROC SQL versus DATA STEP

- SQL uses less steps and is easier to maintain .

- Use DATA Steps and PROC SORT together for large data volumes, whereas PROC SQL is usually better for small data volumes.

Using SAS functions efficiently


```
data work.temp;
length agecat 8;
set source.study1 (keep=pt dob);
if floor((dob-today())/365.25) lt 18 then agecat = 1;
if floor((dob-today())/365.25) ge 18 then agecat = 2;
run;
```

Is much less efficient than:

```
length age 8;
age = floor((dob-today())/365.25);
If age lt 18 then agecat = 1;
else if age ge 18 then agecat = 2;
run;
```

Decreasing I/O

This can be achieved in two ways:

- Reducing the amount of data that needs to be read (These techniques are also used to reduce CPU time)
  - Drop/Keep
  - Where
  - SAS Views (covered in earlier slides)
  - Indexes (will not be discussed)
- Increasing the amount of data that can be read at one time (increased memory usage)
  - Bufno and bufsize option.
  - SASFILE

- Depending on the scenario pick what you see is best .

Decreasing I/O

Quanticate
A PASSION FOR EXCELLENCE

- **Drop/Keep**

```
data work.temp;
set source.study1 (keep=pt dob);
run;
```

- **WHERE**

```
data work.temp;
set source.study1 (where=(vsdt  ne  .));
run;
```

- Decreases CPU time
- Can be used with PROC and DATA step
- *Cant be used to subset on new variables and for conditional processing*
- *IF is more efficient in certain scenarios*

BUFNO and BUFSIZE

Quanticate
A PASSION FOR EXCELLENCE

Syntax:

- The default value for BUFSIZE= and BUFNO = is determined by your operating environment.

- OPTIONS BUFNO=2 BUFSIZE= 16384
  Here 2*16384 =32,768 bytes are transferred in one I/O operation.

- Note:
- BUFSIZE can only be used for output dataset and is a permanent attribute of that data set

- BUFNO is not a permanent attribute of the data set and is valid only for the current step or SAS session

SASFILE

- Syntax and Example:
  sasfile < > OPEN/LOAD/CLOSE;
  where
  - OPEN: opens the file and allocates buffers but defers reading the data into memory.
  - LOAD: opens the file, allocates the buffers, and read the data in memory
  - CLOSE: frees the buffers and closes the file

- Eg: sasfile derived.lb load;

```
proc freq data=derived.lb;
tables cohort*lbtest;
 run;
proc  print data=derived.lb noobs;
 where lbtest= 'ALT' ;
 var subjid lbstresn cohort trt lbunit;
 run;
sasfile derived.lb close;
run;
```

Tips and tricks

- Split your data if possible (eg: 3 datasets for Labs based on labcat)
- Delay deriving variables as much as possible.
- Plan when to execute .

Understanding Trade-Offs

Decreasing	Increased
Data Storage Space	CPU Time, More I/O
I/O	Data Storage Space, Memory, Buffer
CPU	Memory usage
Memory	More I/O
Real time	CPU Time

Summary

- Large datasets affect Data Storage Space, I/O ,CPU and Memory Usage
- Efficient use of simple but powerful tools like Length, Drop/Keep, Where, SAS Functions ,BUFNO, BUFSIZE, SASFILE help address this issue
- Compressing datasets saves I/O and disk space but uses more CPU
- SAS Views can be used to decrease data storage space because they contain no real data, just the code to access them
- Data Normalisation and taking the time to decide on tradeoffs of using PROC SQL versus data step can also reduce data storage space
- Since these factors are interlinked, reducing one will increase the other. So we need to measure the tradeoffs before we decide which method is best

References

- SAS Support, lhttp://support.sas.com/documentation/
- Andrew H. Karp and David Shamlin, SUGI Paper 3-28, Indexing and Compressing SAS® Data Sets: How, Why and Why Not
- Vishal Jain, Quanticate, PhUSE 2010 Paper CC03 :Working Efficiently with Large SAS® Datasets
- Efficient Techniques and Tips in Handling Large Datasets: Shilong Kuang
- www.hollandnumerics.com/SASPAPER.HTM#
- http://www.nesug.org/Proceedings/nesug97/advtut/horwitz.pdf
- Wikipedia

- Many thanks to Bharat Chaudhary, Harsh Gajjar ,Julie Oates.

Contact Info

Gloria D' Souza

Quanticate International Ltd (India Branch Office)
97, Patton House 2, 4th Floor,
4th B Cross, 5th Block,
Kormangala Industrial Estate
Bangalore 560095

Work Phone : +91(0) 8049 110 714

Email : gloria.dsouza@quanticate.com

Web : www.quanticate.com