Paper 4720-2020

Bits and Bytes – A Mix for High Volumes of Data

Karine Désilets, Canada Revenue Agency

ABSTRACT

Building a system is like building a house – both relate to the construction realm and have a myriad of similarities. Handling high volumes of data is like building a tower – or a castle. Every step is critical in order to have a great building! Big towers take sound planning, big cranks and; specific foundations, materials and supplies. Some buildings are trendier than others; some are copied from others; however, we all are familiar of the main plans with so many buildings built in the world.

This paper reviews the roadmap of high volumes of data processing – in a funny fashion! The first part shows how to tackle big data efficiently and presents techniques to reduce the amount of data processed; the second part explains how to reduce the amount of data stored on disks; and the final part demonstrates techniques to efficiently process large datasets. Throughout this article, we expose one principle: Keep it Simple Stupid (KISS) – no need to be complicated. À la mode towers can be down-to-earth!

INTRODUCTION

At Canada Revenue Agency, the philosophy putting people first drives the collaboration and innovation journey. In that context, a technique that is being applied to leverage knowledge sharing and growth on specific topics to other colleagues and team members is through analytical and methodological review and sharing sessions. Those brainstorming sessions are part of the Agency and we also like to call these sessions Bits and Bytes.

Bits and Bytes sessions are commonly used by different personas in analytical related worlds across the industry – They are sometimes referred to with different terminologies such as – Tips and Tricks, Q&A or, simply, tips or advices. Different wording, but the same goal: putting heads together on how to master a topic, sharing ideas, while increasing creativity and productivity amongst peers. Such working environments keep the work interesting and develop great teamwork abilities.

Knowledge sharing within the siloes technical teams has proven to be challenging across the Agency. One of the tricks learned over the years is to harness the power of analytical tools including IBM SPSS, SAS®, R, Python and others. As with any of these languages, it is possible to know a lot about the skills and roles of technical teams by examining their SAS technical level and their programming code.

SAS offers users many alternatives to produce the same results in several different ways. The procedure name selected, the programming options, the programming indentation, the type of comments and even the variables names or the datasets names express a lot about the employee. For that purpose, being early exposed to several SAS coding styles helps incredibly junior programmers to customize their own programming style. In the meantime, when an employee is senior, mentoring and guiding across the functional teams develops their horizontal collaboration. Receiving guidance from a senior employee allows a junior employee to hone their programming skills. Seniors also outshine in helping juniors to develop their vertical relationships with the different level of management.

Another flagship challenge of the Agency is faced with the massive amount of digital data that is continuously acquired, stored and analyzed. At the Agency, there is a pot of gold of current and historical data about several topics of interest such as financial tax data, population, businesses, child

benefits, pension plans, to mention just a few. Having such different ecosystems of knowledge about data, metadata and high data volumes from various business lines increase the complexity of streamlined governance. This is where Bits and Bytes sessions are really useful as they help to consolidate the different domains of knowledge present at the Agency.

To give a bit of history, Bits and Bytes was also the name of a Canadian educational television series produced by TV Ontario in the 1980s-90s. The series taught the basic about how to use a computer. This paper mimics the Bits and Bytes style by using strong analogies, words play and jokes. It highlights various fundamentals topics related to high volumes of data and, with the power of SAS and technical skills, presents easily developed solutions that can be learned quickly.

The essential knowledge regarding handling data volumes including the repetitive topics related to the infrastructure, the data location and the nature of the data are presented. Secondly, the journey of advising about data processing reduction and efficient SAS datasets storage is given and finally, some artifacts about processing large datasets efficiently are presented. Through the article, guidelines about the techniques to process high data volumes with SAS are additionally covered.

HANDLING HIGH VOLUMES OF DATA

Working with high volumes of data can be tricky, scary or sometimes lead to nightmarish situations the minute that all the puzzle pieces are not properly in place or when some major pieces are missing. The more often someone works with the several specific topics related to high volumes of data, the more of an expert that person would become: from the infrastructure, to the programming language syntax, the algorithms complexity or simply, to the number of variables or the number of records (Figure 1). When all the principles are properly understood and applied, handling high volumes of data is also straightforward and should be seen as a recipe – like Mom's outstanding apple pie recipe.

Figure 1. Topics related to high volumes of data

The first element to define is the concept of high volumes of data. The formal definition relates to the concept of Big Data. Gartner® defines big data as high-volumes, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation ([1]). High volumes of data are then governed by the infrastructure such as the availability of memory, storage performance, processing capacity and network speed.

As high volumes of data relate to the infrastructure, the best usage of all these concepts also relates to the concept of high volumes of data for a specific infrastructure. Each individual infrastructure owns a maximum capacity in terms of data processing, storage and performance. Being able to maximally optimize the performance of an infrastructure is tricky and entails the collaboration of several technical specialists; it is state of the art!

THE INFRASTRUCTURE

The notion of artificial intelligence (AI) notion appeared approximately 50 years ago. Since the emergence of AI, many of the mathematical and statistical concepts applied remained relatively homogeneous. During this time, concepts from domain to domain are reapplied with different notations or wording. Where do the most significant changes come from? They come from the IT infrastructure.

The IT infrastructure environment, its different components and usage are decisive. There is no need to have 15 minutes of processing for simple query results: within seconds results can be obtained. If this is not the case, there is a problem. That problem could be with the infrastructure or; it's usage.

There are several important basic notions related to infrastructure. Handling high volumes of data on a laptop? Loading high volumes of data on the laptop GPU card? He.? Ho.? *This is* ~~probably~~ *not a good idea*. Handling high volumes of data on a scalable client-server environment is probably a better computational strategy.

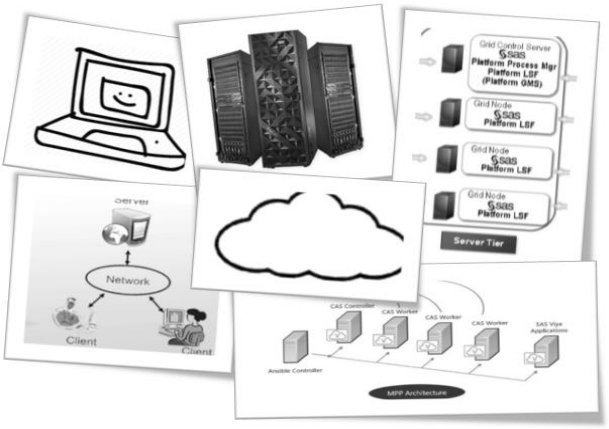


Figure 2. Infrastructure options

As an organization derives value from data, effectiveness of the infrastructure is of vital importance (Figure 2). There are a lot of emerging R&D done on effective storage and server processing architecture - from an efficient server appliance as IBM Pure Data Analytics (PDA Netezza), to SAS Grid infrastructure where the parallelization occurs at the SAS job level or from SAS Viya massive parallel processing (MPP) infrastructure with the in-memory distributed calculus at the procedure level (CAS server). Parallel and pipelines processing environments are often the solutions when handling high volumes of data.

In this regard, SAS® R&D teams are working on creating an efficient combination of a SAS Grid and SAS Viya massive parallel processing installed on the nodes of the SAS Grid ([2]). Wow! The perfect pair! We are really in a hurry to create use cases for that new scalable environment. Will that new parallel environment solve all the current problems of servers' efficiency? Is that new environment, combined with Cloud computing, a better solution? We will have a lot of testing to do in order to ensure these.

THE NATURE OF DIGITAL DATA

The hype about processing and analyzing different types of data is huge. There is a well-kept secret: magical wands and magic do not exist - Santa Claus neither. Processing non-structured data from traditional methodologies is difficult. To meet the requirements of processing all categories of data (Figure 3), the unstructured data need to be transformed into structured form. In a structured dataset, the rows represent the observations and the columns are the features or target variables ([3]).

Type of Data	Description	Examples
Structured	Data defined by data model, format, structure and stored in a repository.	• Transactional Databases • Datasets • Analytical Data Stores
Semi-structured	Textual hierarchical data that contains tags or other markers to separate semantic elements.	• Spreadsheets • Xml Files
Quasi-structured	Textual Data with erratic data format. Often formatted with effort and time.	• Clickstream Data
Unstructured	Data that has no pre-defined data model and not organized in a pre-defined manner.	• Video • Images • Text or PDF documents

Figure 3. Type of data, description and examples

On the perspective of data management, these different types of data force the Agency to innovate and create new emerging ways of processing the digital information received by the taxpayers and businesses. By processing this information, new opportunities are gained and the quality of the services offered to Canadians potentially increases.

THE DIGITAL DATA

Digital data are often available through a variety of sources and are stored at different locations. Computer science historical literature shows that transactional data were first stored in-memory with arrays, tables or pointers. In order to store the data permanently on disks flat files were used and then, the concepts of datasets and databases appear. When transactional data are transformed for analytics, different abstractions clock in.

"Confused about Data Lakes, Data Warehouse and Data Mart?
At the end of the day, IT is all SAS Datasets!"

In the '90's the concept of **Data Mart** took form – having a platform of specific analytic data used by a specific community. Early 2000's, the buzzword was **Data Warehouse** – analytical cleaned and categorical detailed data stored on a centralized repository and often includes data from diverse sources. Since 2010, the trendy word is **Data Lake** – raw analytical data of different non-related topics and of diverse data types stored at the same location.

You are right, the definitions given about data mart, data warehouse and data lakes are not totally fully explicitly given. There exist major differences between data marts, data warehouses and data lakes as stated in ([4], [5]). However, depending on the sources, the definitions are varying. Different definitions but, at the end of the day, these are all SAS datasets ready for reporting, analytics or data scientist explorations.

As a thought leader, the appropriate question we should ask is: what's next? Inter galactic federated data location? This is the type of question that the industry leaders need to examine – Do the data governance models still need to become bigger and bigger? Are we on the edge of adding other types of data or other types of data repository?

PART 1: REDUCING THE AMOUNT OF DATA PROCESSED

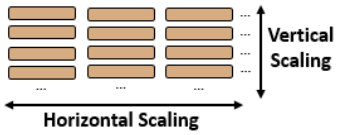
From a technical standpoint, the first technique to apply when processing high volumes of data is to decrease to the minimum number of variables and records to be processed. This can be achieved using KEEP and DROP statements in the SET options of a DATA step.

"When dealing with high volumes of data, only keep the needed ones! The less data you process the more CPU resources you leave to other users."

When high data volumes are present, processing too much information can quickly become unmanageable. Reducing the amount of data processed is the answer. The industry often presents the burden of scaling up as the solution. However, scaling down the amount of data to be processed can also become your best weapon.

SCALE UP - SCALE DOWN

A vertical scaling up refers to adding new data records. In SAS technical language, this is denoted as a PROC APPEND. A vertical scaling down refers to deleting records – it is done with an IF or WHERE statement. The same notion is applicable for a horizontal scaling: adding new variables versus removing variables with a DROP or KEEP statement.



Efficient and effective horizontal scaling down entails removing the irrelevant variables directly in the SET statement. That way, only the needed variables are assigned in the program data vector (PDV)[1] and the other variables present in the original input dataset are not retained. Also, deleting the superfluous final variables with a DROP statement on the output DATA step remains a relatively good practice as shown in Example 1.

```
data scaleDown (drop = income1-income100);
   set historicalIncome (keep = year sin prov age income1-ncome100);
   income = sum (of income1-income100);
run;
```

Example 1. Horizontal scaling down with a KEEP and DROP statements

SUBSET THEM

Another trick to do data selection is to subset the data and consequently, wipe out irrelevant records as early as possible. Remember, a WHERE statement is executed at compilation - only the selected records are loaded into the PDV ([6], [7]). A IF statement occurs at the runtime, the variables are first loaded into the PDV and then, when the IF condition is met the observations are kept (Example 2).

```
data subsetThem;
   set historicalIncome;
   where (2010 <= year <= 2019);
run;
```

Example 2. Vertical scaling down with a WHERE statement

[1] Program Data Vector: The PDV is the logical area in memory where SAS builds a data set, one observation at a time. It is spanned across the compilation and execution phases. The PDV retains the information and the changes that are done during a DATA step.

DEVELOP IN SMALL CHUNKS

When developing SAS programs for high data volumes, do not process the whole database or dataset. Develop with a subset of the records: Process in small chunks! Use selection strategies such as the statement OBS = n (where n is a positive integer representing the numbers of records to keep). Why do we need to apply such techniques? It will be a lot less CPU consuming. Once your SAS code is developed and error-free, then process with all data and perform any algorithmic improvements; as needed.

```
data smallChunk;
   set historicalIncome (obs = 1000);
run;
```

Example 3. Temporary vertical scaling down with a OBS statement

Another way to develop with a sample can be done by using **PROC SURVEYSELECT**. The SURVEYSELECT procedure is a really useful sampling procedure as it returns sampled data that are representative of the population. An example of the procedure, which proportionally allocates the total sample size amongst the strata, is the following:

```
proc surveyselect data = historicalIncome out = smallChunkSampling
   method = srs samsize = 1000;
   strata = prov age / alloc = prop nosample;
run;
```

Example 4. Random sampling with SURVEYSELECT procedure

Using a sample size of 1000 records, Example 4 shows that a stratified random sample is obtained with proportional nonoverlapping strata allocated over the prov (province) and age variables. The NOSAMPLE option specifies that the final sampled records of the input dataset historicalIncome are not assigned in the output dataset smallChunkSampling – Without this option, the final output dataset contains the entire sample of selected units.

Having such a smaller dataset for development of methodology is really useful. However, be careful, do not over subset, process what need to be processed. Data scientists often refer to these sampling strategies as training datasets, test datasets or partition data. It is only a usage of different expressions for the same rearing ideas.

THE INCOMPARABLE APPEND

Adding new observations are part of the process of continuous vertical scalability. With high data volumes it is strategic to have good planning, architecture and clear objectives when adding new observations. Prioritizing processing of new observations and appending new final data to the output datasets should be part of the steps involved. When processing statistical models, that type of processing architecture is often impossible. The addition of observations might require appending observations to the input datasets and reprocessing completely systems or statistical models. There are several options on how that process should be done. Appending new observations is however faster than reading completely input datasets.

When appending data to an already existing dataset there are a couple of options. The most popular options are the DATA step, the APPEND procedure and the SQL procedure. With high volumes of data, the APPEND procedure is much faster as the BASE dataset is not read at all in comparison to DATA step and PROC SQL. However, the APPEND procedure requires that both datasets have the same structure.

Here is a comparison of the three main SAS techniques for appending datasets:

APPEND Procedure	DATA Step	SQL Procedure

`proc append`
`  base = base`
`  data = newData;`
`run;`
	`data base;`
	`  set base newData;`
	`run;`
		`proc sql;`
		`  create table AppDs as`
		`  select * from base`
		`  outer union corr`
		`  select *`
		`  from newData;`
		`  drop table base;`
		`quit;`
		
		`proc datasets library=work;`
		`  change AppDs = Base;`
		`run;`

Figure 4. How to append files with SAS using APPEND procedure, DATA steps or PROC SQL procedure

Furthermore, code readability has to be considered. Which of these three options is easiest to read? Simplicity of maintenance, coding best practices and code readability are also important aspects to consider when selecting a technique. Finally, there is a fourth solution that some prefer to use; this is with the APPEND statements of the DATASETS procedure ([8]) as Example 5 present, below.

```
proc datasets;
   append base = base
   data = newData;
run;
```

Example 5 Appending a file with DATASETS procedure

The APPEND statement of the DATASETS procedure is equivalent to the APPEND procedure in terms of functionality. The only difference resides in some options and where the default libref is for BASE and DATA arguments of both procedures. However, simple appending as displayed above with APPEND procedure and APPEND statement perform identically.

PART 2: REDUCING THE AMOUNT OF DATA STORED ON DISKS

High volumes of data involve high capacity of storage, which can become quickly costly. Having A governance for data storage is elemental but erasing unneeded datasets and avoiding duplication of datasets are essential. These are simple and straightforward strategies. However, making sure that people follow correctly these strategies is challenging. Everyone has their own idea of how to do it. Ensuring uniform application is mostly impossible. In these cases, go back to basics and apply the following tricks in order to reduce data storage volumes.

KEEP THAT LENGTH UNDER CONTROL

How should disk space usage be reduced? There are several techniques, but correctly managing the length of SAS variables present in datasets is the simplest. SAS assigns default length values to character and numeric variables. By reducing the length of these variables, you can save disk space. The default numeric variable length is 8 and it can be reduced down to 3. Similarly, the first value assigned in a dataset determines the length of characters variables, and the subsequent larger values are truncated. Characters variables length can span from 1 to 32767 bytes. Also, remember that for numeric variables a length of 3 bytes, on Unix, can represent a maximum numeric value of 8,192 while a length of 8 bytes can represent numbers up to 9,007,199,254,740,990. You may not need all those numbers. Allocating a length of 8 bytes might be required.

There are two techniques to assign the number of bytes when storing variables on disks. Both techniques are done in a DATA step (Figure 5). The first one assigns the length with the statement LENGTH and the second one, with the LENGTH option of the ATTRIB Statement. Which one should be used? The main point is that the LENGTH statement only assigns length, where as in an ATTRIB statement each individual attribute such as length, label, informat and format can be assigned.

ATTRIB Statement	LENGTH Statement

`data historicalIncome;`
`  attrib varn1 varn2`
`  varn3 length=3;`
`  attrib varc1 varc2`
`  length=$4;`
`  set historicalIncome;`
`run;`
	`Data historicalIncome;`
	`  length varn1 varn2 varn3 3`
	`  varc1 varc2 $4;`
	`  set historicalIncome;`
	`run;`

Figure 5. Reducing the length of variables within a DATA step with the ATTRIB or LENGTH statement

Note that Flag variables that have value of 0 or 1 are typically defined as numeric and, occupy 3 bytes. Using character variable, with length of 1, reduce the number of bytes to one. This is demonstrated by the variable IncomeFlag in Example 6 below.

```
data flagDs(drop=var1);
   length IncomeFlag $1.;
   set historicalIncome;
   if (Income >= 100000) then
      IncomeFlag = 0;
   else IncomeFlag = 1;
run;
```

Example 6 Assigning flag variable with a character variable of 1-byte length

SQUEEZE THAT DATASET

The **%SQUEEZE** macro, created by Ross Bettinger, can find the minimum lengths required by numeric or character variables for a SAS data set and use these lengths to reduce the size of the dataset. The source code is available at http://support.sas.com/kb/24/804.html. There is no need to assign variables lengths manually.

TO COMPRESS OR NOT TO COMPRESS

In each records of a dataset, variables have a specific length. Since all records of a dataset occupy the same amount of storage regardless of whether there are any data in the record, compression is then the process to reduce the number of bytes involved in representing each observation.

Compression is good for datasets that are Character intensive or Binary intensive. As the goal is to reduce; the number of bytes that each observation requires can then be lessen with the COMPRESS DATA step option or the COMPRESS system option. Be careful; it is not free – some CPU cycles are involved in order to compress. Compression can be done with the system option COMPRESS will compress all datasets created during a SAS session. Alternatively, compression within a data step compresses each individual dataset (Figure 6).

Compress with the System Option	Compress with a Dataset Option

`options compress = binary;`
 	
`data mydata (compress = yes);`
`  set work.historicalIncome;`
`run;`
 	
`data mydata (compress = no);`
`  set work.historicalIncome;`
`run;`
	`data mydata(compress = char);`
	`  set work.historicalIncome;`
	`run;`
	
	`data mydata(compress = binary);`
	`  set work.historicalIncome;`
	`run;`

Figure 6. Compressing Datasets with System OPTIONS or in a DATA Step

Note that compression is an essential detail when having large datasets (over millions of records), large character variables and many numeric variables. The compress algorithm with option BINARY gives better results for dataset having mainly numeric variables and conversely, the option CHAR gives better results on characters variables. It might be prudent to try both options in order to determine which one, of the two, gives the best results.

Most of the time, compression is required when the data have: common patterns, empty spaces and numeric variables ([9]). In presence of small datasets, with few variables and few repetitions, compression is disabled as shown in the SAS log below:

```
NOTE: Compression was disabled for data set WORK.MYDATA because compression
overhead would increase the size of the data set.
NOTE: There were 1 observations read from the data set WORK.HISTORICALINCOME.
NOTE: The data set WORK.MYDATA has 1 observation and 1 variable.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
 cpu time            0.01 seconds
```

Figure 7. SAS Engine disables compression with small datasets

SUCH A NICE VIEW

In computer science, a view is a virtual table that is defined by a query that is retrieving all the information from different data files. Instead of creating a new data file (dataset), the data sources location, the variables and the descriptor information such as data types and length of variables are saved into the view. A view is pretty useful as it reduces data storage volumes. However, as usual, some CPU cycles are required in order to retrieve all the data. A view saves disk space but increases processing time – There is no perfect solution.

When having high volumes of data with frequent access to a view, processing the view increases a lot the I/O in order to retrieve all the information from the different data sources. However, a view is really useful when the observations change frequently as a view read the data sources each time the view is executed.

PROC SQL View	DATA Step View

`proc sql noprint;`
`  create view myViewSQL as`
`  select * from  historicalIncome`
`  where income ne .`
`  order by prov, sex, sin;`
`quit;`
	`data myView1`
	`  (sortedby = prov sex sin) /`
	`  view = myViewDs;`
	`  set historicalIncome;`
	`  where income ne .;`
	`run;`

Figure 8. Creating SAS view with SQL procedure or with DATA Step

There are two ways to create SAS views: with a DATA Step or with a SQL procedure. The APIs for both options are pretty similar as shown in Figure 8 and the outcome has exactly the same resulting data file.

DELETE ALL OF THEM?

Servers' storage management with high volumes of data is very important; the last adequate strategy to use in order to reduce the amount of data stored on disks is to delete datasets themselves. Deleting the datasets take some processing time and not deleting the datasets take some disks space. What is the solution? There is a need for good compromise and guidelines.

Deleting a SAS dataset as soon as it is not used clears up the SAS data libraries and frees some disks space. Additionally, a SAS session has a quota of Workspace and Utility space associated to the session. With high volumes of data, those two specific quotas could be easily reached – You can increase those spaces but it is much better to delete the datasets that are no longer mandatory. Again, the secret is to delete the transitional datasets at the end of a macro and to minimize the creation of new datasets.

There are several approaches to delete datasets:

PROC Delete	`proc delete;`
	`  data = mylib.datasetName;`
	`run;`

PROC Dataset and Delete Statement	`proc datasets lib = mylib;`
	`  delete datasetName1;`
	`  delete datasetName2;`
	`run;`

PROC SQL and Drop Table	`proc sql;`
	`  drop table mylib.datasetName1;`
	`  drop table mylib.datasetName2;`
	`quit;`

SAS Enterprise Guide and Conditional drop	`%eg_conditional_dropds(mylib.datasetName);`

Deleting and De-assigning a whole SAS Library	`%macro deleteLibrary(libIn = );`
	`  %if %sysfunc(exist(&libIn.))  %then %do;`
	`  proc datasets lib = &libIn. kill nolist nodetails;`
	`  quit;`
	`  %end;`
	
	`  %if %upcase(&libIn.) ne WORK %then %do;`
	`  libname &libIn. clear;`
	`  %end;`
	`%mend deleteLibrary;`
	
	`%deleteLibrary (libIn = Mylib);`
	`%deleteLibrary (libIn = work);`

Figure 9. Deleting datasets with the different SAS Procedures

The first three option of Figure 9 show formal SAS code to delete datasets themselves. The conditional drop with SAS Enterprise Guide, deletes the dataset only if the dataset exists; if not, there is no drop procedure executed. The macro %deleteLibrary for deleting a whole library is pretty useful for deleting a whole library and the datasets: the WORK library or a library previously created by the user.

PART 3: PROCESSING LARGE DATASETS EFFICIENTLY

Processing data implies building a system, an analytical process or a data scientist exploration. From data gathering to data visualization, each step is critical and is related to construction – construction of systems or visualization mechanisms. In that regard, the IT people are the skilled craftsmen, the data is the stockpile of building materials and supplies and the algorithms are the powerful tools that can help to construct a valuable structure from that stockpile ([3]).

Dealing with high volumes of data is like building a tower. Avoiding unnecessary steps, DATA Steps and reports are important in order to accelerate the processing. However, it takes time and good plans to build great buildings. The plans for the building depend on the design and are unique to every building dream up.

One of the leading blueprints in computing practices is to first use a pipeline process, also known as a data pipeline. By connecting the flow of processes in series, where the output of one element is the input of the next one and by parallelizing the tasks - when the infrastructure allows parallel processing - there will be a lot of performance gain.

On the other hand, using proper SAS procedure is more accurate than developing your own SAS algorithm. When a SAS procedure or a SAS function already exist, take advantage of them. SAS took care of optimizing them – there is no need to reinvent the wheel. Although, also remember that the processing time also depends on the infrastructure. If the proper infrastructure is not in place, you might not have major speediness gains even if you have the most optimized SAS code.

Effective and robust scalable plans, infrastructure, algorithms and specialists are required when handling high data volumes. Plans are often copied or improved from past building construction, and some buildings are built in series in a federated environment. However, ability to provide simplified processes and approaches is often fundamental when thriving with international standards of building systems.

A HUGE NO TO O(6N)

Notation O(f(n)) is a mathetical notation. It is called notation Big **O.** Big **O** notation describes the performance and the complexity of an algorithm as the dataset size grows. Big **O** notation describes also the worst-case scenario of the execution time required by an algorithm. The parameter n is referred to as the “size of the problem”. The function f(n) can be read as the time it takes to solve a problem of size n. Basic examples of problems of linear complexity size **O**(n) and of quadratic complexity size **O**(n^2) are the following:

O(n)	**O(n^2)**

`x = 1;`
`for i = 1 to n then do;`
`  x = x + i ;`
`end;`
	`x = 1;`
	`for i = 1 to n then do;`
	`  for j = 1 to m then do`
	`  x = x + i + j ;`
	`  end;`
	`end;`

Figure 10. Examples of problems of size n and size n^2

On the left side of Figure 11 below, each DATA Steps is a linear problems of size **O**(n). The SAS code example show a function f(Cn) where C = 6. That problem is still a problem included in **O**(n). However, when handling high volumes of data, the denoted constant C means a lot. Each time the dataset is read, it takes a certain amount of processing time ([7]).

Reading the dataset C times takes an amount of Cn minutes. Reading the dataset once and applying all the calculations or processing sequentially in one dataset augment the performances. Sometimes, it is not possible to create all the calculations or processing in one dataset but a good dosage of DATA Steps creation is often appropriate.

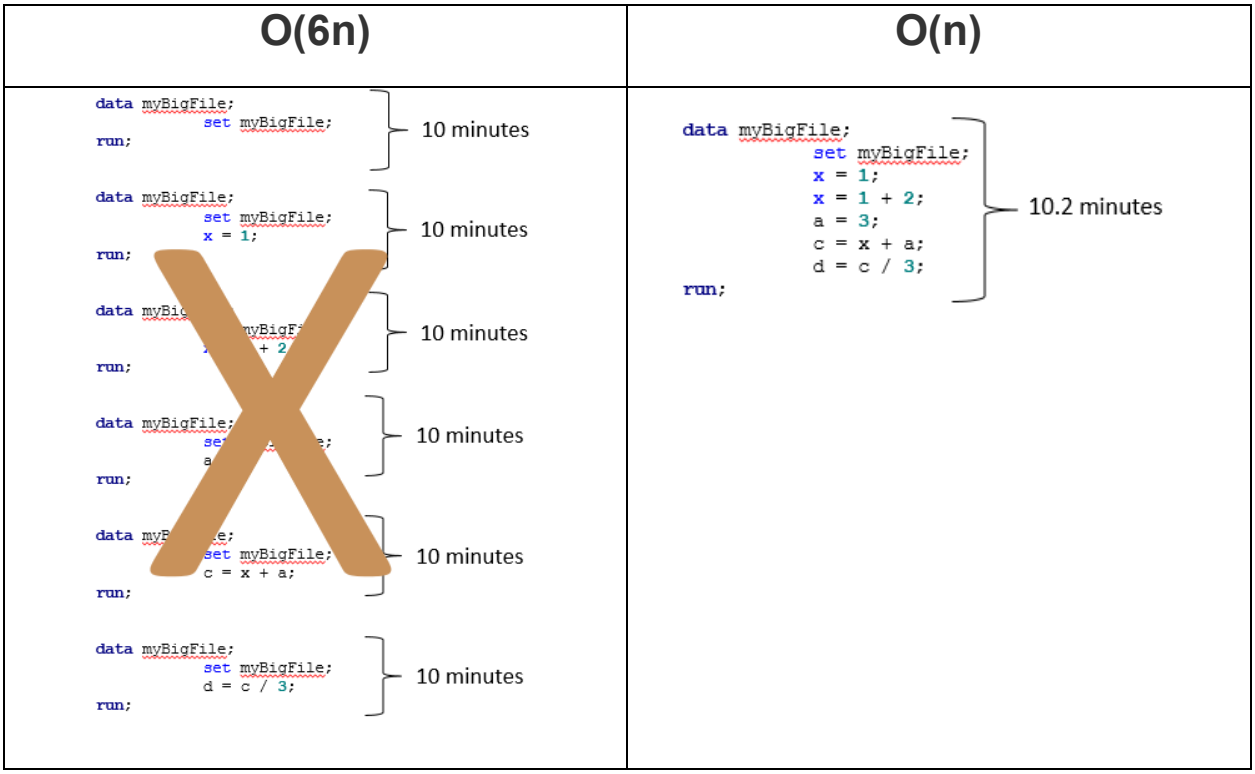


Figure 11. A huge NO to O(6(n))

Building efficient algorithms order is fundamental with high data volumes. However, remember that the majority of the efficiency in the algorithms has already been done – The efficient "wheel" for each situation have been created long time ago . You still want to improve the speed? Most of the time you will need to play with the infrastructure performances options as SAS environment automatic macro variables and other SAS different options.

DATA STEPS OR SQL PROCEDURES? GIVE IT A TRY.

When the time comes to select DATA Steps or SQL procedures there is a lot of turmoil about which of the two techniques performs the best, process the fastest or which API offers the simplest interfaces. There are many arguments on the topic from a broad range of individuals. Who has the correct answer? It is often difficult to say, but keep an open mind. There is no perfect answer, sometimes the winners are DATA Steps; sometimes the winners are SQL procedures. The answer also depends on the characteristics of the infrastructure where you execute the SAS code. Most of the time, interchanging both approaches is recommended – some tasks are easier with SQL procedures and sometimes tasks are easier with DATA Steps. There are several papers about that subject ([6], [10], [11], [12], [13], [14], [15]) - just to name a few. Note that SQL procedures are not supported on the CAS server of SAS Viya.

The SAS DATA Step's first release was in the beginning of the 1970's - SAS Institute was incorporated in 1976. IBM researchers created Structured Query Language (SQL) in 1975 a decade before PROC SQL was included in SAS version 6.0 – or near version 6.0. The implementation of SQL in SAS is not

compliant to ANSI SQL standards; some features have not yet been implemented. Furthermore, SAS has adapted some features of DATA Steps intentionally for PROC SQL.

The major difference between both is that DATA Steps process sequentially. DATA steps respectively loads the data in a buffer of records, build the processes in the PDV, outputs sequentially the data manipulations in the output buffer, and outputs the final results in the output data set. Distinctively, SQL procedure first scans the query, and the optimizer intelligently determines how to execute the query and the different data manipulations in-memory ([15]) before outputting the resulting dataset. When dealing with a high volumes of data and PROC SQL, a high capacity of in-memory infrastructure is mandatory.

DATA Step Simple Merge	PROC SQL Joint

`proc sort data = inDs1 out = outDs1;`
`  by prov sex sin;`
`run;`
 	
`proc sort data = inDs2 out = inDs2;`
`  by  prov sex sin;`
`run;`
 	
`data dataStepMerge;`
`  merge outDs1 (in = A )`
`  outDs2 (in = B);`
`  by prov sex sin;`
`  if A and B then output;`
`run;`
	`proc sql _method ;`
	`  create table sqlMerge as`
	`  select A.*`
	`  from inDs1 as A`
	`  inner join inDs2 as B on`
	`  A.prov = B.prov and`
	`  A.sex = B.sex and`
	`  A.sin= B.sin;`
	`quit;`

Figure 12. Comparison of DATA Step merge and PROC SQL inner jointure

When concatenating tables, each DATA Step to be merged needs explicit sort before merging, and the SET operator can handle more datasets at a time than PROC SQL outer joins. With PROC SQL, the sort manipulations are handled behind the scene. See Figure 12 for an example of simple merge between two datasets and a PROC SQL equivalent inner jointure. Both examples use alias, A and B, to refer to the datasets themselves.

When analyzing PROC SQL with the _method option we notice that some data manipulations are hidden from the API but are still processed behind the scene.

```
NOTE: SQL execution methods chosen are:
      sqxcrta
          sqxjm
          sqxsort
              sqxsrc( WORK.INDS1(alias = A) )
              sqxsrc( WORK.INDS2(alias = B) )
NOTE: Table WORK.SQLMERGE created, with 428 rows and 19 columns.
39         quit;
```

Figure 13. SAS log output of PROC SQL inner jointure with _method option applied

Figure 13 shows that Sqxrc read data sources INDS1 and INDS2; secondly, sqxsort applies a sort operation on the data sources; thirdly, sqxjm merges the sorted data sources and finally, the output table is created with the step sqcrta. We could conclude that PROC SQL is an expert in hiding the real data manipulations done as PROC SQL language declare only the manipulations to be outputted – not how it will be done. In comparison, DATA Steps always explicitly declare the operations to be processed.

There are a lot of comparisons that can be done between DATA Steps and PROC SQL. Exploring both methods are highly recommended and choosing the preferred methodology. Also, when having high volumes of data, the SQL Pass-Through Facility might be better. That facility subsets the data directly on the database management system (DBMS) before the data are transferred to SAS datasets. By doing so, the SQL Pass-Through Facility would reduce the I/O processing time – particularly when the DBMS is an high performance environment.

INDEXES, INDEXES, INDEXES

One of the first goals when having massive amount of data is speeding up the search for information within the datasets. SAS indexes would greatly improve the search for variables values amongst large datasets with a lot of records. Indexes are specifically designed for having a direct access to specific observations that are not frequently present in a voluminous dataset. The more frequent a variable value is, the less optimal an index would be. A good example of good indexes usage is on a variable as Social Insurance Number (SIN).

SAS index provides a direct access to specific observations by creating a physical ordered data file containing variables values and the location of those values within the observations in the primary datasets associated to an indexed data file ([9], [16], [17]). An index can be composed of:

- One unique key variable (simple unique key index)
- Many unique key variables (composite unique key index)
- One or many non-unique key variables (simple or composite indexes)

As shown in Figure 14, there are three ways to create indexes in SAS: with PROC Datasets, with DATA Step or with PROC SQL. The same resulting indexes are created in each code example. A unique index on variable SIN, a non-unique index on variable PROV (province) and a composite non-unique index on variable PROV and CITY also referred under the index name GEO. The three different APIs provide the same SAS implementation of indexes.

PROC SQL	`proc sql;`
	`  create unique index sin on historicalIncome;`
	`  create index Province on historicalIncome;`
	`  create index geo on historicalIncome(Province, City);`
	`quit;`

PROC Datasets	`proc datasets library = work;`
	`  modify historicalIncome;`
	`  index create sin / unique;`
	`  index create Province;`
	`  index create geo = (Province City);`
	`run;`

DATA Step	`data work.historicalIncome (index = (sin / unique`
	`  province`
	`  geo = (province city)`
	`  )`
	`  );`
	`  set work.historicalIncome;`
	`run;`

Figure 14. Indexes creation with PROC SQL, PROC Datasets or with a DATA Step

Multiple indexes can be created against the same data file. An Index speeds up the location of records in the data file and when indexed properly, no data file sorting is required. However, creating indexes add some CPU time and require disk space as a physical file is created. When an index is used for search, it also requires some additional buffers memory. An Index takes CPU and disks resources for adding or deleting observations, and Indexes might increase the I/O cost in some situations. Finally, indexes are not required for small data files – in particular when a dataset has less than a three pages count. With low page count, it is faster to access data sequentially ([17]).

Extraction with an Index	SAS Log

`options msglevel=i;`
 	
`data Extract;`
`  set historicalIncome;`
`  where province = "Quebec";`
`run;`
	`32  options msglevel = i;`
	`33  data Extract;`
	`34  set historicalIncome;`
	`35  where province = "Quebec";`
	`INFO: Index province selected for WHERE clause optimization.`
	`36  run;`

Figure 15. How to see if an index is used or not when sub-setting dataset

When extracting data from a dataset having an index, the option msglevel = i has to be activated in order to have a INFO note in the SAS log saying whether or not an index has been used for search (see Figure 15).

JUST PASSING THROUGH

The PROC SQL Pass-Through Facility allows you to send SQL statements directly to a database management system (DBMS) rather than being executed by PROC SQL ([7], [18]). Data movements are minimized when the data are first subset by the DBMS before being sent to PROC SQL. A good example of SQL Pass-Trough is the interaction of SAS with IBM Pure Data Analytics (PDA Netezza). SAS has developed a SAS/ACCESS interface to Netezza. At the Agency, the data lakes for our data are on IBM PDA Netezza appliances.

When data are transferred to SAS servers from IBM PDA Netezza appliances, I/O processing time occurs. In order to reduce the I/O processing time, we use the PROC SQL Pass-Through Facility. There are two types of connection to DBMS, an explicit one with the SQL pass-through and an implicit one with the LIBNAME statement (IPASSTHRU option).

Explicit SQL Pass-Through	Implicit SQL Pass-Through with Libname

`proc sql passtrough;`
`  connect to netezza as db`
`  (server =  XXXX`
`  database = XXXX`
`  authdomain = XXXX);`
`  execute (`
`  create table IncomeData as`
`  (`
`  SELECT *`
`  FROM IncomeData`
`  )`
`  ) by db;`
`quit;`
	`LIBNAME NZ netezza`
	`  server = 'XXXX' Database = 'XXXX'`
	`  schema = XXXX authdomain = XXXX;`
	
	`proc sql ipassthru;`
	`  create table IncomeData as`
	`  select *`
	`  from NZ.IncomeData;`
	`quit;`
	
	`data IncomeData_ds1;`
	`  set NZ.IncomeData;`
	`run;`

Figure 16 Explicit and implicit connection to IBM Pure Data Appliance

When comparing both APIs of SQL pass-through (Figure 16), we discovered that for each SQL query, the connection string to the server needs to be asserted at each explicit Pass-Through declaration. However, with the libname and the IPASSTHRU option, the connection string is declared once, and then the libname is referred through a libref. The libref and IPASSTHRU option are a shorthand way, an alias, for executing the pass-trough when having multiple data extractions. For both explicit and implicit pass-through, the SQL query occurs on the DBMS and the result is transferred to the SAS server.

Also, as IBM Netezza appliance is compliant to SQL, the interface for extracting data is more powerful with SQL procedure. There are a lot of functions that are not compatible with the DATA step. However, the implicit pass-through option with a libname and DATA step with a libname are available most of the time with the different DBMS.

Dataset Income (100,000,000 rows and 17 columns)	**Execution time**	
Explicit SQL Pass-Through	real time	7:28.44
	cpu time	7:27.10
Libname and SQL implicit	real time	7:29.48
	cpu time	7:27.79
Libname and DATA Step	real time	7:51.75
	cpu time	7:18.72

Figure 17. Processing time comparisons of communication between SAS and IBM Pure Data Appliance.

Figure 17, shows that the execution times are relatively the same and performance is consistent while DATA step show some slower execution time where real time and CPU are not consistent. Note that DATA step example has been processed at different time of day and always showed comparable execution time results. In that specific test case scenario, DATA step is less efficient than SQL procedure.

CONCLUSION

Within different industries, the concept of high data volumes have exponentially grew in popularity and data lakes of structured, semi-structured, quasi-structured and non-structured data shape the actual digital economy. Working with high volumes of data involves harnessing the power of advanced analytical tools with an alignment of the infrastructure – Cloud or not. From complex infrastructures as SAS Viya Massive Parallel Processing (MPP), Hadoop Cluster to any other infrastructure design as IBM Pure Data Appliance (PDA) or powerful SAS servers.

This article has strategically explored diverse thematic in order to identify the core elements required to better position the actual SAS artifacts to process high volumes of data. The article also gave you some guidelines to the actual computer science roadmap of processing high volumes of data. In order to address the gap between high volumes of data processing and appropriate infrastructures, final metrics related to data rough size, data growth factor, data retention, CPU demand and CPU usage have to be calculated.

Leveraging previous investments made to the infrastructure in order to optimize the actual infrastructure is also part of the continuum of handling high volumes of data. However, best practices of server's efficiency involve reducing the amount of data processed to the minimum, reducing the amount of data saved on disks and ensuring that large datasets are processed efficiently.

The limitation of each topics presented has been discussed and the solutions presented are not perfect. Even with a scalable platform there is a continuous dilemma contained by the topic of: disks space versus CPU consumption. Defining a final vision and governance in order to strengthen and ties the different topics together should also be part of future initiatives. However, as per today, simple actions should be part of each programmer priorities:

- ✓ Deleting irrelevant datasets or avoiding redundant datasets creation
- ✓ Not reinventing the wheels and using the already optimized built-in solutions, macros, procedures and functions
- ✓ Programming intelligently and effectively while keeping the solutions with the flavor of Keeping it Simple, Stupid (KISS Principle)
- ✓ Ensuring a good balance of elegance and functionality with the SAS programming
- ✓ Thinking of SAS Program maintenance
- ✓ Using performance techniques related to SAS DATA steps, PROC SQL, PROC DS2 or Viya In-memory processing
- ✓ Making certain to understand the infrastructure capabilities and characteristics
- ✓ Using Google or other web research engine as your best weapon or friend

The number of variables and the historical amount of data assorted to the complexity of the interrelations between variables, variables topics and subject matter related increase the analytical complexity of high data volumes. When putting-in innovation as a priority, interdisciplinary flexible teams are often required and ensure that the emerging solutions are on the edge of Excellency when dealing with high volumes of data issues. Moving forward with the above addressed topics is also fundamental and lead to success. Some broader explorations would have to been made in order to explore other options such as the impact of a robust specialized workforce in presence of high data volumes versus infrastructure financial plans.

As a final point, empowering new technologies and high data volumes manipulations are giving more capabilities, lead to further alternatives and shape the road of the current society. However, are these new technologies and high data volumes manipulations can harm us? When crossed with interconnected topics of technologies, ethics and human rights these theoretical and empirical explorations are part of the work of many worldwide academics. Furthermore, many individuals, companies and governments have been involved in ethical controversies in the past few years. Some have paid the price and for several; there is a sword of Damocles hanging over them. Formal analysis of these topics should be part of further articles.

REFERENCES

[1] "Gartner Glossary," https://www.gartner.com/en/information-technology/glossary/big-data.

[2] SAS, "SAS® Grid Manager and SAS® Viya®: A Perfect Pair," SAS, https://www.sas.com/en_us/webinars/sas-grid-manager-sas-viya-a-perfect-pair.html, 2019.

[3] B. Wujek, P. Hall and F. Güneş, "Best Practices for Machine Learning Applications," in *Paper SAS2360*, AS Institute Inc., 2016.

[4] B. Botelho, "Data warehouse vs. data lake vs. data mart: Beyond the RDBMS," TechTarget, [Online]. Available: https://searchdatamanagement.techtarget.com/feature/Beyond-the-RDBMS-Data-warehouse-vs-data-lake-vs-data-mart.

[5] P. Simon, "Data lake and data warehouse – know the difference," SAS, [Online]. Available: https://www.sas.com/en_us/insights/articles/data-management/data-lake-and-data-warehouse-know-the-difference.html.

[6] S. Agarwal, "The Secret Life of DATA STEP," Paper BB-03-2013, Eden Prairie, 2013.

[7] J. Novotny and S. M. Bramley, "We All Do: Mistakes We've Made That You Won't Have To," in *MWSUG-2010*, Cary, NC, 2010.

[8] "Base SAS(R) 9.2 Procedures Guide - APPEND Statement," SAS, https://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000235213.htm, 2019.

[9] R. Landge, "SAS Techniques for Managing Large Datasets," in *PharmaSUG*, Cary, NC, 2016.

[10] F. Dosani and H. Droogendykb, "Joining Data: Data Step Merge or SQL?," in *Proceedings of the SAS Global Forum*, Cary, NC, 2008.

[11] F. J. Malachy, "MERGING vs. JOINING: Comparing the DATA Step with SQL," in *SUGI 30*, Cary, NC.

[12] C. Dickstein and R. Pass, "DATA Step vs. PROC SQL: What's a neophyte to do?," in *SUGI 29*, Cary, NC.

[13] K. P. Lafler, "Top Ten SAS® Performance Tuning Techniques," in *MWSUG*, Cary, NC, 2012.

[14] "SAS® 9.4 SQL Procedure User's Guide, Fourth Edition," SAS, NC, [Online]. Available: https://documentation.sas.com/?docsetId=sqlproc&docsetTarget=p07h5bmiljefenn1qmibhfcbx5c9.htm&docsetVersion=3.2&locale=en.

[15] J. D'Agord, "SAS Coding: PROC SQL Versus DATA Step," Zencos, 2016. [Online]. Available: https://www.zencos.com/blog/sas-data-step-vs-proc-sql-superbowl/.

[16] M. A. Raithel, "Creating and Exploiting SAS® Indexes," in *SUGI 29*, Montréal, 2004.

[17] SAS , "Understanding SAS Indexes," SAS, [Online]. Available: https://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a000440261.htm.

[18] S. Feder, "The LIBNAME Engine Compared to SQL Pass-Through," Washington, DC, 2007.

[19] J. McGuinn, "tutorials4u," [Online]. Available: http://tutorials4u.com/c-tutorial/bits-bytes-number-systems.htm.

[20] S. First and T. Schudrowitz, "Arrays Made Easy: An Introduction to Arrays and Array Processing.," in *SUGI 30*, Cary, NC.

[21] L. Palmer, "Using SAS Views and SQL Views, State of California," in *WUSS*, Richmond, CA, USA, 2003.

ACKNOWLEDGMENTS

Thanks to SAS Global Forum 2020 related teams who selected the abstract. It is always a great pleasure to be part of the SAS Community. Thanks to Analytical Consulting Unit team members for advising on typos and related. Thanks to Family: your support is always highly valued.

RECOMMENDED READING

- Artificial Intelligence at The Canada Revenue Agency, A Plan to Move Forward, December 2019

- Sheila Jasanoff, Pforzheimer Professor of Science and Technology Studies, Harvard Kennedy School https://sheilajasanoff.org

CONTACT INFORMATION

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

Karine Désilets
Canada Revenue Agency
613-670-7356
karine.desilets@cra-arc.gc.ca
https://www.canada.ca/en/revenue-agency.html