PharmaSUG 2024 - Paper AP- 289

Programming with SAS® PROC DS2: Experience with SDTM/ADaM

Jianfeng Wang, Graduate Student, Biostatistics, University of Minnesota Twin Cities / 2023 Summer Intern, Vertex Pharmaceuticals Inc.;
Li Cheng, Principal Statistical Programmer, Vertex Pharmaceuticals Inc.

ABSTRACT

PROC DS2 is a procedure introduced with SAS Base 9.4. This procedure provides opportunities for SAS programmers to apply Object Oriented Programming (OOP) and multithread techniques in SAS programming and is a critical connection between 'traditional' SAS programming and programming in the SAS Viya® platform. The goal of this paper is to pilot the use of PROC DS2 in the work of preparing clinical trial CDISC datasets. In this paper, PROC DS2 is tested in the programming of SDTM/ADaM on a server with SAS Base 9.4 M3 release. After converting SDTM/ADaM programs written in 'traditional' SAS programming language into the PROC DS2 code, this paper presents the lessons learned and the notes taken when the obstacles were overcome or bypassed. Furthermore, OOP and multithread techniques are explored to apply to the programming for SDTM/ADaM. Programming setups with a standard folder structure are discussed and the performance of using OOP and multithread techniques are also evaluated.

INTRODUCTION

PROC DS2 is a key advancement in SAS Base 9.4, which is also supported by the SAS Viya platform (SAS Institute, 2013). PROC DS2 is the next generation of the data step in SAS, which enables the data step to use powerful SAS PROC structure and functionality. With PROC DS2 as a bridge, SAS programmers can access more advanced programming techniques, such as Object-Oriented Programming (OOP) and multithread. This paper focuses on applying PROC DS2 in preparing SDTM/ADaM datasets for clinical trials. SDTM/ADaM datasets are fully introduced in the book listed in the references (Case and Tian, 2022). Transitions from 'traditional' SAS programming to PROC DS2 are discussed together with the challenges and practical solutions in programming with PROC DS2.

The paper is structured into the following parts: review of PROC DS2's Object-Oriented Programming structure, multithreads in practice, application in preparing SDTM/ADaM datasets, and PROC DS2 challenges and solutions in SDTM/ADaM Programming. Each section aims to provide the details of hands-on experience in using PROC DS2 to program for SDTM/ADaM datasets.

REVIEW OF PROC DS2'S OBJECT-ORIENTED PROGRAMMING STRUCTURE

PROC DS2 allows programmers to define packages (classes in OOP context). Attributes and methods can be defined within the package. Objects, or instances, can also be created based on package. Here is a quick example:

```
proc ds2;
    package person();
        dcl char(20) self_name;
        dcl double self_weight;
        dcl double self_height;
        method set_attributes(char(20) name, double weight, double height);
          ...
        end;
        method CalculateBMI() returns double;
          ...
        end;
    endpackage;
run;
quit;
...
```

```
proc ds2;
    data data1;
        dcl package person subject();
        method run();
            set data0;
            subject.set_attributes(name, weight, height);
            bmi = subject.calculateBMI();
        end;
    enddata;
run;
quit;
```

Other OOP concepts, such as inheritance and polymorphism, can also be implemented with PROC DS2. PROC DS2 also allows programmers to define data types and create user-defined methods. Further details about the OOP features of PROC DS2 are discussed in the papers listed in references (Serhii, 2017 and Mazloom, 2017).

MULTITHREAD IN PRACTICE

The performance of multithreads with PROC DS2 in various scenarios has been carefully studied in the paper listed in reference (Hughes, 2019). We conduct our own multithread test of complex process with 2-layer do-loops on a SAS 9.4 M3 server with 8 CPU cores. The dataset for the test is a large dataset created from repeating “sashelp.cars” 1000 times (428,000 observations). Here is the code structure:

```
proc ds2;
    thread testThread;
        dcl double i j k;
        ...
        method run();
            set work.cars;
            ...
            do i = 1 to 100;
                do j = 1 to 50;
                    k = (j*i)**3;
                end;
            end;
        end;
        method term();
            ...
        end;
    endthread;
run;
quit;
...
proc ds2;
    data cars3/overwrite=yes;
        dcl thread testThread t;
        method init();
            ...
        end;
        method run();
            set from t threads=1; ** test threads=1 to 16 here **;
        end;
    enddata;
run;
quit;
```

The number of threads is tested from 1 to 16. The performance of multithreads (measured in real time) is improved with the number of threads from 1 to 8. However, after the number of threads reaches 8, we observed a slight performance decrease with number of threads going up. Our conclusion is that the maximum performance of multithread is reached when number of threads equals the number of CPU cores. The results of our multithread are summarized in the Table 1 below.

Table 1: Performance for different thread numbers

Threads	Real Time	CPU Time
1	11.37s	11.60s
2	6.20s	12.70s
4	3.37s	13.57s
8	2.73s	18.93s
12	2.90s	18.87s
16	2.87s	18.67s

APPLICATION IN PREPARING SDTM/ADAM DATASETS

PROC DS2 is still a relatively new tool for SAS programmers working in clinical trials. In order to gain hands-on experiences, we apply PROC DS2 to SDTM/ADaM programming through the following steps:

- Convert 'traditional' SAS data steps into PROC DS2 as much as we can.
- Use multithread to improve the efficiencies in complex processes, normally seen in ADaM while handling the endpoints with complicated calculations.
- Apply OOP in appropriate scenarios.

The experiences gained from above activities together with lessons learned are explained in the later section.

For packages and threads needed in PROC DS2, the corresponding library folders are created at the same level as macro library folders, for example the folders look like below at reporting level:

- ..*Compound_folder**study_folder**reporting_folder***packagelib**\
- ..*Compound_folder**study_folder**reporting_folder***threadlib**\

PROC DS2 CHALLENGES AND SOLUTIONS IN SDTM/ADAM PROGRAMMING

In the process of converting 'traditional' SAS code into using PROC DS2, the first important lesson we learned is that:

- A PROC cannot be used in another PROC in SAS

Therefore, we focused on using PROC DS2 to replace data steps in SDTM/ADaM programming. A typical PROC DS2 has a structure of the data declarations followed by init() method, run() method and term() method. Here is what a complete PROC DS2 may look like:

```
proc ds2;
    data data1;
          dcl char(5) var1;
          dcl double var2;
          method init();
                ...
          end;
          method run();
                set data0;
```

```
            ...
        end;
        method term();
            ...
        end;
    enddata;
run;
quit;
```

PROC DS2 provides various data types available to choose from. In comparison, the 'traditional' SAS data steps only have two data types by default, string and double. The data declaration is normally the first hurdle to deal with while start using PROC DS2. The run() method is the implicit looping similar to a 'traditional' SAS data step that loops through the records one by one in a dataset. The init() method and term() method are two one-time data manipulation opportunities before and after the run() method. In general, the conversion from 'traditional' SAS data steps to PROC DS2 can be as simple as copying data step logic into run() method. In addition, the following PROC DS2 features are quite helpful in our SDTM/ADaM conversion efforts:

- User-defined method is a powerful tool, which has helped a lot in replacing the SAS functions that won't work within PROC DS2. Once a user-defined method is defined, it can be used in the same way as SAS function. A typical user-defined method (*user_defined*() in the sample code below) looks like the following:

```
proc ds2;
    data data1;
        dcl char(5) var1;
        dcl double var2;
        method user_defined(double flag1) returns char(5);
            ...
        end;
        method run();
            ...
            var1=user_defined(var2);
        end;
    enddata;
run;
quit;
```

- SQL statements can be directly used in PROC DS2 without calling PROC SQL.
- Macro used in data steps could continue to work in PROC DS2. If the macro has the programming parts mentioned below, further modifications may be required.

However, the following programming parts commonly used in SDTM/ADaM programming may encounter issues while converting to PROC DS2:

- Merging two datasets works differently in PROC DS2. In SAS Base 9.4 M3, it is pretty hard to achieve the same merging results in PROC DS2 as SAS data steps. There is a /RETAIN option available in SAS Base 9.4 M6 release to solve the issue. For our test in SAS Base 9.4 M3, we just keep merging in a 'traditional' SAS data step instead of moving it into PROC DS2.
- Some SAS functions commonly used in SDTM/ADaM programming may not be available in PROC DS2, such as *input*(), *cmiss*() and *compress*(). To mitigate the impacts, we use user-defined method to create replacements for some functions, such as *input*() and *cmiss*(). For complex functions like *compress*(), it is hard to create a replacement and we simply use them in a 'traditional' SAS data steps instead of using PROC DS2.
- Some date formats, such as *is8601dt.*, cannot be recognized in PROC DS2. In PROC DS2, there are two functions *todate()* and *todouble()* that can help in mitigating the date format issue.

- The default encoding of input datasets may cause a warning message in log file while batch run the SDTM/ADaM programs using PROC DS2. In order to remove the warning message, the encoding of the input datasets could be specified as 'wlatin1' before being processed by PROC DS2. Here is one example:

```
libname inlib cvp (work);
libname outlib '.\temp' outencoding='wlatin1';

proc copy noclone in=inlib out=outlib;
      select ...;
run;
```

Other notable differences frequently seen, while applying PROC DS2 in SDTM/ADaM programming, are:

- SAS autocall macro function %tslit() should be used to call a macro variable in a string.
- Logic expression is less flexible in PROC DS2, see examples below.
- String with double quotation marks is not allowed in PROC DS2, instead single quotation marks should be used for string.

Examples of above differences are show in the following table:

Not allowed in PROC DS2	Allowed in PROC DS2
`domain='&domain.';`	`domain=%tslit(&domain.);`
`if aeout=:'NOT' then do;`	`if aeout like 'NOT%' then do;`
`if aestdt ne . then do;`	`if aestdt ^= . then do;`
`"string"`	`'string'`

Multithread processing was tested while applying PROC DS2 in SDTM/ADaM programming. Even though the performance improvement is evident in complex calculation processes as shown in the earlier part of this paper, performance improvement from multithread is hard to notice in simple data manipulations, especially when input from files and output to files are heavily involved. The order of datasets created from multithread may be unpredictable due to different threads potentially having different speeds on different CPU cores. Therefore, extra steps may be needed to keep datasets in intended order under multithread environment.

Some features of OOP are also tested while applying PROC DS2 in SDTM/ADaM programming. Even though OOP has a lot of potential in SDTM/ADaM programming, a lot of work is needed before programmers can fully enjoy the full benefits from OOP. Here is a summary of what are needed for fully exploiting the power of OOP:

- Redesign SDTM/ADaM programs from OOP perspectives. For example, design a generic SDTM package for finding domains and then create SDTM finding domains through inheritance and polymorphism.
- More standard packages are provided from SAS (similar standard library in C++ programming or packages available in R programming) for SAS programmers to start programming with PROC DS2.
- SAS is more OOP friendly, for example, by providing an OOP friendly editor to allow SAS programmers to easily program with OOP features in PROC DS2.

CONCLUSION

PROC DS2 provides ways to augment current SAS programming with the capability of OOP and multithread. Some PROC DS2 feathers may immediately help current SDTM/ADaM programming, such as handling complex calculations with multithread on multi-core computer and creating user-defined methods. However, in longer run, the potential benefit of PROC DS2 will be maximized when SAS programmers redesign current SDTM/ADaM programs from OOP perspective, more packages are available for use and the SAS programming environment is more OOP friendly.

REFERENCES

Case, Todd and Tian, YuTing. 2022. *An Introduction to Creating Standardized Clinical Trial Data with SAS®*. Cary, NC: SAS Institute Inc.
https://support.sas.com/content/dam/SAS/support/en/books/an-introduction-to-creating-standardized-clinical-trial-data-with-sas/78769_excerpt.pdf

Hughes, Troy Martin. "User-Defined Multithreading with the SAS® DS2 Procedure: Performance Testing DS2 Against Functionally Equivalent DATA Steps". Proceedings of the Annual Pharmaceutical SAS Users Group Conference, 2019.
https://www.pharmasug.org/proceedings/2019/AD/PharmaSUG-2019-AD-228.pdf

Mazloom, Dari. “Object-Oriented Programming With SAS® Via PROC DS2”. SCSUG, 2017.
https://www.scsug.org/wp-content/uploads/2017/10/dm82.pdf

SAS Institute, Inc. 2013. SAS® 9.4 DS2 Language Reference.
https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/ds2ref/titlepage.htm

Serhii, Experis Clinical, Kyiv, Ukraine. "Expansion of Opportunities in Programming: DS2 Features and Examples of Usage Object Oriented Programming in SAS®". Pharma SUG, 2017.
https://www.lexjansen.com/pharmasug/2017/BB/PharmaSUG-2017-BB09.pdf