PhUSE 2014

Paper AD02

How to improve performance on SDD 3.5

Rob Wartenhorst, Novartis Vaccines, Amsterdam, Netherlands

ABSTRACT

Novartis Vaccines moved from a legacy SAS ® environment to SAS Drug Development (SDD 3.5). With the move the data format was converted from the legacy data structure to SDTM data structure and the reporting macros updated to use the SDTM data. To get a head start on the conversion, the reporting macros were updated on the legacy SAS environment. Not realizing how the repository moves its data around we ended up with a very inefficient use of the system. This paper addresses a technique we applied to make system performance more acceptable.

INTRODUCTION

At Novartis Vaccines we were using a windows based SAS environment. For automated complience with current and future regulatory requirements such as automated version control and access control for external parties, SAS Drug Development version 4.0 was selected as the future platform. As version 4.0 was not yet available, the planned release data was a few months in the future, we started rewriting our programs on the windows platform assuming that SAS = SAS and only minor changes would be required, to make the programs work in the new system. Rewriting the SAS programs was required because at the same time we also moved from our legacy data format to SDTM.

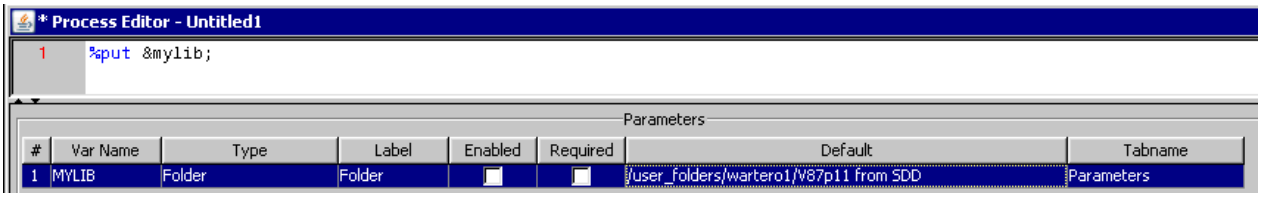
As version 4.0 was not released yet we did not have access to a sandbox environment to learn about the system. Time passed and version 4.0 release was delayed with a year for a major rewrite due to performance issues and instead we were given version 3.5 of SDD instead.

PROGRAM SETUP

A typical program setup in our environment consists of an analysis program and a setup file. This setup file sets up the input libraries/folders, output libraries/folders and the format catalog plus global macro variables. This setup file is included by every analysis program. We have a folder for input datasets, a folder for SAS programs, .lst and .log files, a folder for output pdf files and a folder containing meta data (read only) and itemstore files (write).

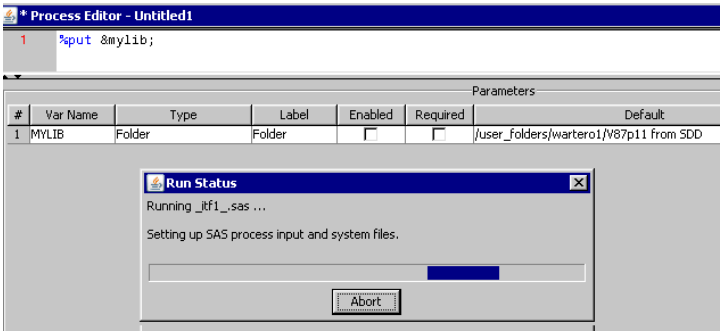
HOW THE REPOSITORY WORKS WITH FILES

Simply said, there are 2 servers. One server on which the files are stored and one server on which the files are processed. You always work with a copy of the file (unless you access the data using WebDAV, however WebDAV is not stable). You get access to a file in a folder by defining a folder through a macro variable.



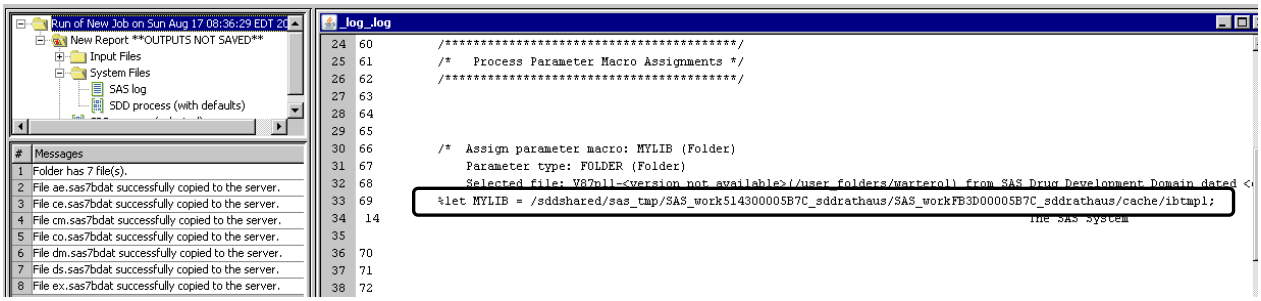
1. Defining a folder though a macro variable.

When the code is executed, the files are copied from the repository to the workspace. You can't really see what is happening. You just have to wait.



2. Executing folder assignment

Once the program completes, the messages window indicates which files have been copied and in the log you can see the physical location of the folder to which the files have been copied from the repository. We call this the workspace.



3. Files are available in workspace

In SDD 3.5 a folder can be set up as one of the following 3 types of folders:

1. an Input folder,
2. an output folder
3. or an input/output folder.

When you define an input folder the files from this folder are placed in the workspace. When you define an output folder a file location is created in the workspace. Important! If you create an input folder and an output folder that using the same folder in the repository, they will be assigned their own folders in the workspace. So for a SAS program using the folders, the folders are separate, while in the user's mind these folders are the same file location in the repository. This means that you cannot read a dataset from the input folder that has been updated in the output folder while being in the same SAS session.

After the program completes, the following happens for each type of folders:

1. Files from input folders are discarded.
2. Files from the output folder are committed back to the repository.
3. For an input/output folder every file in the workspace is compared with the same file in the repository to check for changes and only changed files are committed back to the repository.

In summary, every time a program is executed, temporary folders are created, files are copied to the repository, and then output files plus .lst and .log files are copied back to the repository. I.e. reading a file involves copying the file from the repository and then reading in by the SAS program. Similar saving files, first involves writing the files to the workspace and then copying the files to the repository. This overhead is noticeable, but with an intelligent set up of folders kept to an acceptable delay.

THE EFFECT OF WINDOWS STYLE SETUP ON PERFORMANCE

- Input folders – Imagine a pooled analysis with 1GB of input data, 300 program files plus .lst and .log files and a folder set up to read the input data and a folder set up to include a file from the directory where the programs are located. This means that 300 GB + 300*300*3 files will be copied when all programs are run. Not only that. If these programs are all run as part of one job, then these files will not be deleted until the job completes. A very large workspace is required.
- Output folders – Output folders do not have an effect on the performance, other than that files are saved to the workspace first and then copied to the repository.
- Input/output folders – Setting up a folder as input/output triggers SAS to check every file in that folder after program completion for changed files. Imagine again the study with 300 programs with 300 output files, then we will have 300 files in this input/output folder. This means 300 times 300 files comparing when all programs are run.

The overall effect was that running a single program could take between 1 and 15 minutes depending on the size of the datasets. Even running a single datastep in a program has the system resolve the folder variables and coping over the data without actually assigning a library to the copied data. Performance statistics indicated that very little CPU was used on the servers and that upgrading hardware would not help improve performance.

IMPROVED APPROACH

When you have 2 or more process editors open in SDD, then they all share the same SAS session. They share the same WORK library and global macro variables. Hence the advice that in principle only one process editor session is active so that programs do not interfere with each other. However this behavior can be used to our advantage as well. We create 2 setup files.

- One file that defines all the input folders. We call that the initialization file.
- One file that defines the output folders plus global macro variables. Note an output folder has to be defined for each program as files are only committed to the repository if an output folder is defined within the program.

When developing the analysis programs, the program file that defines all the input folders is executed once as the first program to be executed and kept active in a minimized state (keeping the session active so you can open and close TFL programs without losing the folders and library assignments). All other programs then can utilize the same input folders.

The setup file that defines the output folders and global macro variables is included as was done in the original setup of the programs.

When running the programs in batch, the setup file that defines the input folders is to be defined as initialization task

Input/Output folder type should be avoided since this type needs to be set up in each program run and results in copying over all input datasets and then checking all copied files for changes, even if only one file was added or changed.

DISADVANTAGES

- User needs to remember to run the initialization file before running the program the user intends to run.
- If accidentally closing all programs in a session, programs don't run and initialization needs to be run again. This can lead to some frustration.
- If input data is updated, user needs to remember to run initialization file again to make the updated data available.
- Implementing this approach after all standard macros have been validated resulted in some situations in a clunky solution. E.g. when a libname is hard coded in multiple validated macros where in one macro its only purpose is to write files and in another macro to read files. To avoid an input/output folder, the libname has to be redefined for the specific purpose of the program which results in not so easy to follow library assignments.

UNEXPECTED CONSEQUENCES

The bottleneck for performance in the initial set up was the network traffic between the server on which the data was stored and the server on which the programs are executed (the application server). As the application server was waiting for data most of the time, the load on this server was quite low (and therefore SAS had said that adding more CPU power was not needed). Immediately after applying the improved approach, the load on the application server became high, very high. Batch jobs now in fact were able to put CPU usage between 80% and 100%. The jobs completed much faster, but the system became quite unresponsive for interactive users. As a consequence a second application server had to be installed to bring system performance back to acceptable levels.

CONCLUSION

The improved approach cuts the run time of a final run of all programs in half. It allows better use of the application server, allowing more hardware to be used. After the second application server was installed a final run of all programs would be completed in a quarter of the original run time, of course given the same load on the system. Developing code on SDD is quite acceptable when there is no data that needs to be copied from the repository to the workspace for every run of the program. Running a single data step can go as quick as 6 seconds. There are now even programmers who don't mind working on SDD.I personally really like the built in log checking. If we would have fully understood how the data flow worked in SDD before we implemented and rolled out the solution, we would have had a significantly less frustrated user base. At the other hand, now the users are happy with the improved situation and do not complain as often about performance which it still is slower than the legacy system, which is mainly due to the overhead of the system having to copy files from and to the repository.

CONTACT INFORMATION

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

Rob Wartenhorst
Novartis Vaccines
Hullenbergweg 81-89
1101 CL Amsterdam
The Netherlands
Work Phone: +31 205640564
Email: Rob.wartenhorst@novartis.com

Brand and product names are trademarks of their respective companies.