



How to improve performance on SDD 3.5

Rob Wartenhorst

Phuse Annual Conference 2014

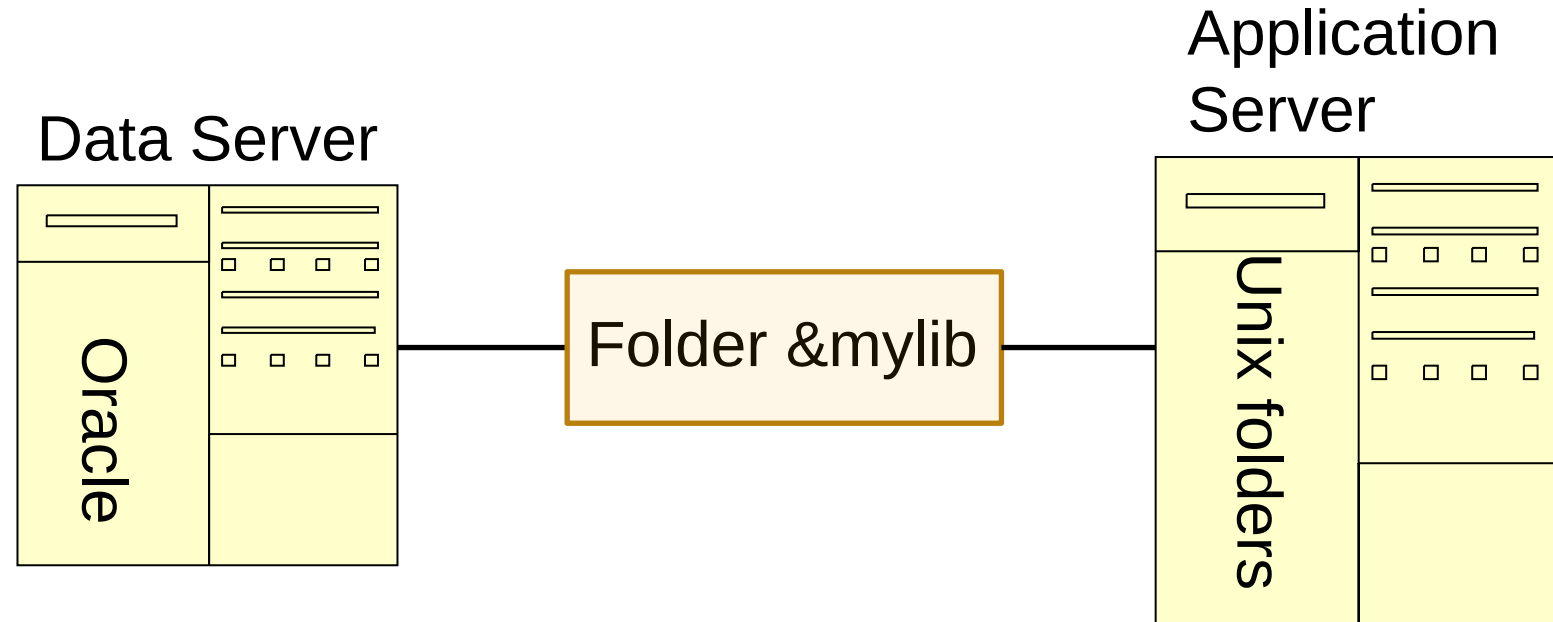
London, October 2014

Introduction

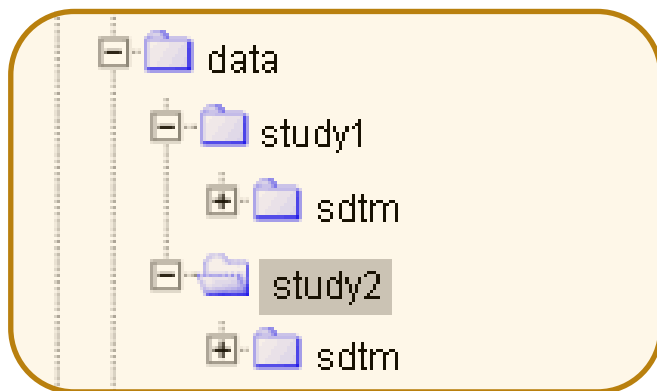
- NVD implemented SAS Drug Development (SDD) as a data repository with automated tools for compliance with regulatory requirements.
- Migrating to SDD included migrating to SDTM datasets and a rewrite of library of standard programs.
- Due to delay in release of version 4, version 3.5 was installed instead.
- Sandbox environment available late.
- Rewriting SAS programs was done on our legacy server.
- Very little exposure to new environment before rollout.

How the repository works with files

Physical setup of servers (simplified)



Presentation in browser



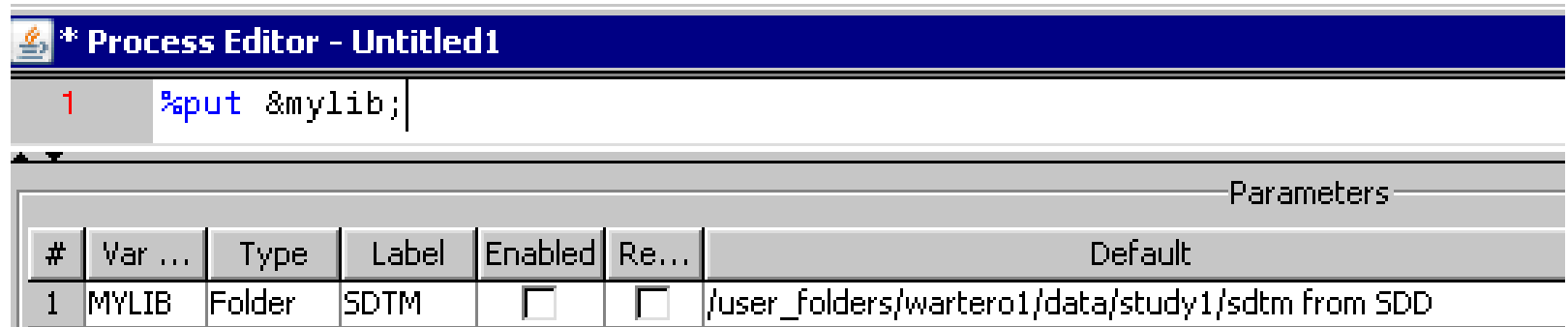
Folder on application server

```
/sddshared/sas_tmp/  
SAS_workD0BF00004A69  
_sddrathaus/SAS_work  
134500004A69_sddrathaus  
/cache/ibtmp2
```

How the repository works with files

Processing Folder macro variables

■ Folder definition



■ First program run

```
261      %put &mylib;  
/sddshared/sas_tmp/SAS_workDOBF00004A69_sddrathaus/SAS_work134500004A69_sddrathaus/cache/ibtmp2
```

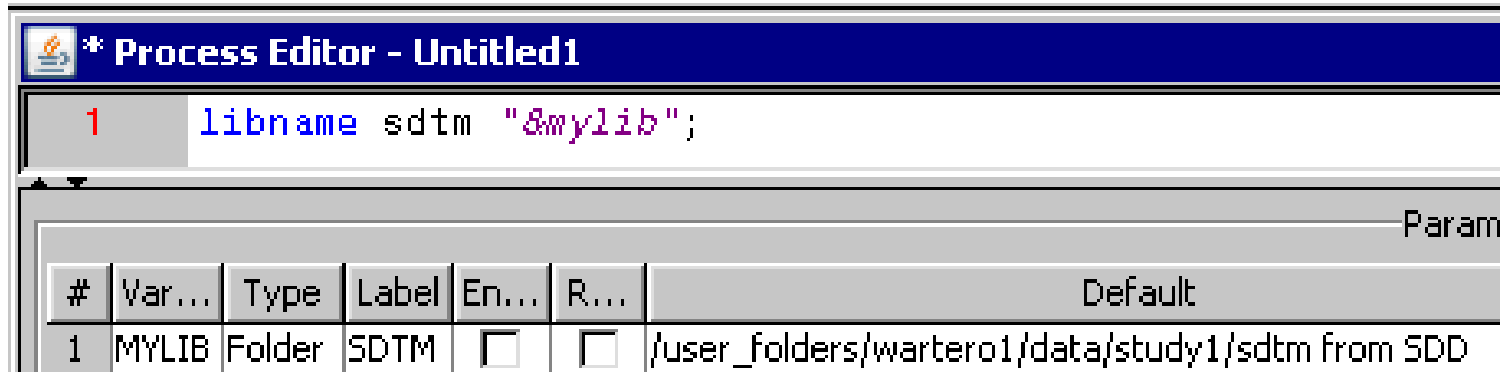
■ Nth program run

```
573      %put &mylib;  
/sddshared/sas_tmp/SAS_workDOBF00004A69_sddrathaus/SAS_work134500004A69_sddrathaus/cache/ibtmp10
```

How the repository works with files (input folders)

Processing Folder macro variables

- When assigning a folder to a libname you can access the data



- SDD assigns the unix folder to &mylib

```
/* Assign parameter macro: MYLIB (SDTM)
   Parameter type: FOLDER (Folder)
   Selected file: sdtm-<version not available>(/user_folders/wartero1/d
%let MYLIB = /sddshared/sas_tmp/SAS_workDOBF00004A69_sddrathaus/SAS_work
```

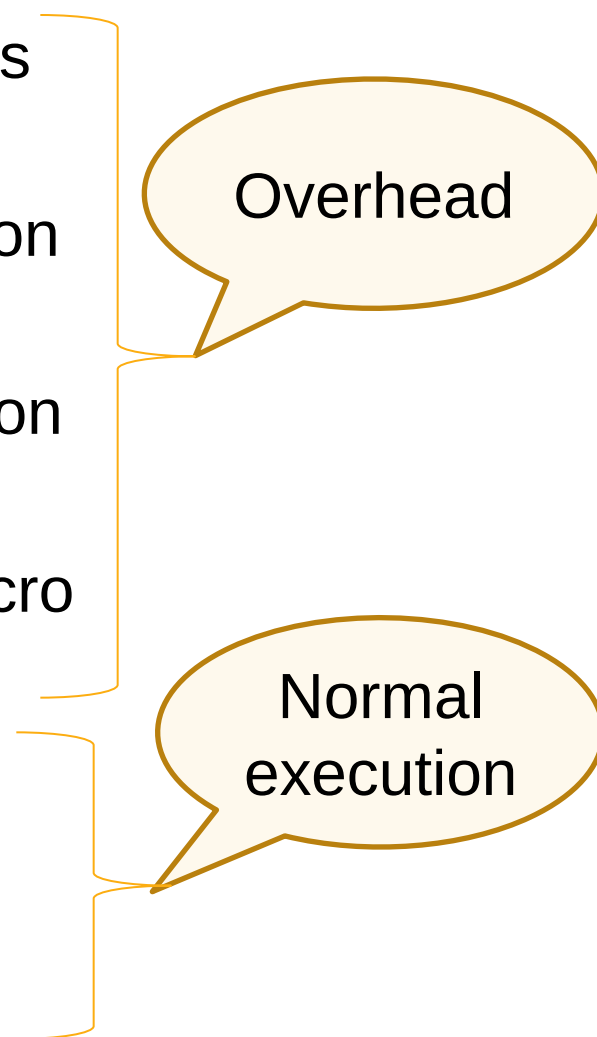
- In the program we can now access a copy of the data

```
libname sdtm "&mylib";
Libref SDTM was successfully assigned as follows:
Engine:          V9
Physical Name:   /sddshared/sas_tmp/SAS_workDOBF00004A69_sddrathaus/SAS_wo1
```

How the repository works with files

What happens when you access a dataset

- SDD scans program for macro variables assigned as folder parameter.
- For a folder, create a temporary folder on application server
- Copy data from data server to application server
- Assign temporary folder location to macro variable assigned as folder parameter
- Assign libname to macro variable assigned as folder parameter
- Access dataset in datastep



Overhead

Normal execution

Reducing overhead

Background info

- Multiple process editors share the same SAS session.
- All programs in a job share the same SAS session.
- A libname defined in one program still exists in another as long as at least one process editor is open.

Reducing overhead

The solution

- Create 2 setup files.
 - One initialization file that sets up the input folders
 - One setup file that sets up output folders and global macro variables.
- All TFL programs only include setup file with output folders and global macro variables as input process (special type of include file).

Reducing overhead

During development of TFL programs

- Run initialization program first, keep active in minimized state.
- Develop/Run TFL program 1
- Develop/Run TFL program 2
- Develop/Run TFL program 3
- Etc...

Reducing overhead

In Batch jobs

- Make sure the initialization program is the first one that is executed.
- Include in the job, all the programs that utilize the same input data.

File Edit Tools Help			
Clinical Data View:			
Initialization Task: _init_read_folders.sas from RELATIVE(../sas/_init_read_folders.sas)			
#	Name	Label	
7	lae001.sas	lae001.sas	../sas/lae001.sas
8	lae006.sas	lae006.sas	../sas/lae006.sas
9	lae007.sas	lae007.sas	../sas/lae007.sas
10	lce001.sas	lce001.sas	../sas/lce001.sas
11	lcm001.sas	lcm001.sas	../sas/lcm001.sas

Folder types

Definition

■ 3 types of folders

- Input folders
 - Data is copied from the data server to the application server. After program completes data is discarded.
- Output folders
 - Empty folder is set up on the application server. After the program completes files written to the folder are committed to the data server.
- Input\Output folders
 - Data is copied from the data server to the application server. After the program completes any new or changed files are committed to the data server.

Folder types

Example

Physical files

\study1\adam
adsl.sas7bdat
adae.sas7bdat

Libname in '\study1\adam';
Libname out 'study1\adam';
Libname inout 'study1\adam';

Data available in 'in':

Adsl.sas7bdat
Adae.sas7bdat

Data available in 'inout':

Adsl.sas7bdat
Adae.sas7bdat

Data available in 'out':

Empty folder

Folder types

Output folders

- Have to be set up in each program (or through included input process) as otherwise files are not committed to data server
- No performance gains possible through smart setup.
- Performance of output folders is not too bad
- Data written to a location in the data server through output folder in program A can not be read by program B if they are run in the same session (or batch job for that matter).

Folder types

Input/Output folders

- Have to be set up in each program (or through included input process) as otherwise files are not committed to data server
- Data written to a location in the data server through input/output folder in program A can not be read by program B if they are run in the same session (or batch job for that matter).
- Quite heavy performance penalty due to:
 - Data will be copied for each program run as folder has to be defined within program
 - All files copied from data server are checked for updates before committing back to the server.
- If at all possible do not use. This folder type decreases performance significantly.

Effect on performance

- Programs run in seconds instead of minutes.
- Programs were not waiting for data any more, increasing CPU use of application server from around 10% to between 80% and 100%. Installation of a second application server was now justified.
- For a very large integrated analysis with many long programs, this technique cut running of all programs in half (before second application server was installed) from 24 hours to less than 12 hours.

Disadvantages

- Users need to remember to run initialization file first (not so straight forward to run a program)
- Initialization program needs to be rerun if accidentally closing all programs in a session.
- If input data is updated, initialization program needs to be rerun to make updated data available. Users sometimes forget this and continue to work with outdated data.
- Implementing this approach in the current set of standard programs results in sometimes a clunky solution.

Summary

- Create initialization file that defines all input folders
- The initialization file should not be included as input process, but run by itself as first program for a SAS session.
- Output folder assignment is still to be done in the program itself (or through an input process).
- Do not use input\output folder type

Conclusion

- SDD 3.5 is not so bad after all. Some programmers even start to like it with the improved performance, although slower than the legacy system, but paperless!
- When implementing a new environment, reserve more budget/time for performance impact analysis before rolling out the new environment.
- Be prepared that by taking away an IO bottleneck, a hardware upgrade may be required.
- A system with electronic tracking of the files will always be slower than an open system. But having the system do much of the documentation does make that part of the job more tolerable.

Any Questions?

