Paper 1721-2014

Deploying a User-Friendly SAS® Grid on Microsoft Windows

Houliang Li, Frederick, MD

ABSTRACT

Your company's chronically overloaded SAS environment, adversely impacted user community, and the resultant lackluster productivity have finally convinced your upper management that it is time to upgrade to SAS grid to eliminate all the resource problems once and for all. But after the contract is signed and implementation begins, you as the SAS administrator suddenly realize that your company-wide standard mode of SAS operations, i.e., using the traditional SAS Display Manager on a server machine, runs counter to the expectation of SAS grid – your users are now supposed to switch to SAS Enterprise Guide on a PC. This is utterly unacceptable to the user community because almost everything has to change in a big way. If you like to play a hero in your little world, this is your opportunity. There are a number of things you can do to make the transition to SAS grid as smooth and painless as possible, and your users get to keep their favorite SAS Display Manager.

INTRODUCTION

In the pharmaceutical company where I work as a SAS administrator, there were two SAS environments: a production server and a validation server. They both ran on Windows 2008 R2 and had attached local disks for data storage. Our SAS programmers, statisticians, and other SAS users typically connected to the production server via Remote Desktop Connection and launched the SAS Display Manager for all their development, testing, and production needs. There were also huge batch jobs scheduled to run on the server for clinical data management. To solve the chronic resource contention issues and better prepare for future growth, we purchased the SAS grid software. The plan was to upgrade from single-server SAS 9.2 to grid-based SAS 9.3.

In preparing for the upgrade, we realized that SAS grid actually favors SAS Enterprise Guide as the user interface. All the grid enablement features are built-in within SAS EG 5.1; checking a couple of boxes is all you need to go to the grid. Therein lay our essentially insurmountable problems. With the exception of a few users who had utilized SAS EG on our legacy UNIX systems, most of our SAS programmers did not know how to use SAS EG. This was at best a small learning curve, but we had a much bigger hurdle. The overwhelming majority of our SAS processing, be it for development, testing or production, was tightly integrated with our unique Windows file system structure tailored for clinical studies. The standard procedure for launching SAS was to use a Windows Command Prompt, type a command to set the environment, and then launch SAS with another command. The two commands guaranteed that each SAS session would start in the right folder, could access the correct files and libraries, and would produce various kinds of output in the right locations. Our scheduled batch jobs used a similar approach. Clearly, using SAS EG could not easily duplicate this successful model.

There were other obstacles as well. Many of our SAS programs directly invoked Windows commands and batch scripts to prepare raw data, such as unzipping input files received from vendors. They also generated PDF files by calling Adobe Distiller and RTF / DOC files via DDE code. SAS EG cannot handle these tasks even with the **`xcmd`** option turned on. Had we decided to go with SAS EG, many of our existing SAS programs would have to be updated, and even broken into separate pieces, before the same tasks could be accomplished in the new and supposedly "better" grid. It would have been an absolute nightmare!

GRID-INSPIRED CHALLENGES

It was an easy decision to stick with our faithful / fateful SAS Display Manager, but we still faced a number of challenges on our march to the grid. The immediate concern was providing user credentials for validating with the metadata server. The official SAS guide book, ***Grid Computing in SAS 9.3***, provides this example for using SAS Display Manager with the grid:

```
options metaserver='metadata-server-address';
options metaport=metadata-server-port;
options metauser=username;
options metapass="password";
%let rc=%sysfunc(grdsvc_enable(grid&count, server=SASApp));
signon grid&count;
```

Clearly, all users are expected to hardcode their passwords into their SAS programs. This is a big no-no in any corporate environment. Alternatively, the **`metapass`** option can be left blank, and the user will be prompted for password for each new SAS session. Even though the password can be remembered for the rest of the session, this approach does not work for scheduled batch jobs. Another related issue is updating the user password for the Platform middleware. All users must use the **`lspasswd`** utility in a Command Prompt to provide their passwords to LSF and later update them when changed.

The second problem was the production file system on Windows. The SAS grid requires a shared storage location so that all grid nodes can access the same files and folders. This ensures that whatever grid node you are directed to by the SAS Grid Manager can run the program successfully. In a single server environment, this is not a problem at all, whether the file system is a local disk or a network drive. As soon as the user logs on to the server, the drive is available (a network drive can be mapped by the logon script). However, the network drive in a grid environment works entirely differently. While the user and his or her local SAS session can indeed have access to the network drive immediately upon logon to the grid control server, the grid session, which is launched by the Platform middleware on a grid node via SAS/Connect, does not see any network drive. It is in essence a batch process that involves no interactive logon, so the network drive cannot be mapped with a logon script. Without access to required files and folders, the grid session cannot do much!

Then there was the inconvenience of dealing with two SAS sessions simultaneously. There is a small learning curve with SAS/Connect for sure; very few SAS users have ever used the product before. Users would be required to add **`rsubmit`** and **`endrsubmit`** statements to send code to the grid for execution. This meant all production code might need to be changed to at least embed these two statements. In SAS Display Manager, our programmers might also need to assign the same libraries in both the local and grid sessions – there are certain tasks that only the local SAS session can handle, such as using DDE to work with Word or Excel. In addition, since each of the two SAS sessions has its own WORK library, it would be ideal to have access to the grid session's WORK library from the local session. When you run code that generates temporary data sets in the grid session, don't you want to examine them and verify that your code is correct?

Finally, the default behaviors of SAS grid, especially how the **`sasgrid.cmd`** file launches grid sessions, conflicted with our established practices of using study-specific **`autoexec`** files, specifying initial folders, and selecting default printer drivers. To make matters worse, every time SAS was launched from an access-controlled study folder or subfolder to use the grid, an empty "work**"** folder would mysteriously pop up right there and would stay forever, in total violation of our corporate policy regarding production environment file system management.

Any one of those challenges could potentially derail our SAS grid deployment if not properly addressed. Unfortunately, ***Grid Computing in SAS 9.3*** has a clear bias in favor of the UNIX operating system and often ignores the Windows platform. It has limited utility to us, or other Windows SAS grid users. On the other hand, the SAS Technical Support department, particularly its grid support team, has been consistently helpful throughout our testing and deployment process. But in the final analysis, we had a unique, highly customized SAS environment that, when upgraded to SAS grid without fundamental modifications, would sharply deviate from what SAS grid expects. We had no choice but to come up with our own solutions.

OPPORTUNITY ONE – HIDE AND FORGET ABOUT PASSWORDS

To reconcile using a password and protecting it, we created a batch utility that every user must run when they first log on to the grid control server via Remote Desktop Connection. This program sets up a Windows environment variable in the user's work space and stores an encoded version of the user password as produced by the SAS procedure PWENCODE. It also automatically feeds the user password to the **`lspasswd`** utility which then stores it in an encrypted form for the Platform middleware. When a user changes his or her password once every three months per

the corporate security policy, a logon script automatically detects the inconsistency, opens up a Command Prompt and instructs the user to run the same batch program again. See **Figure 1**.

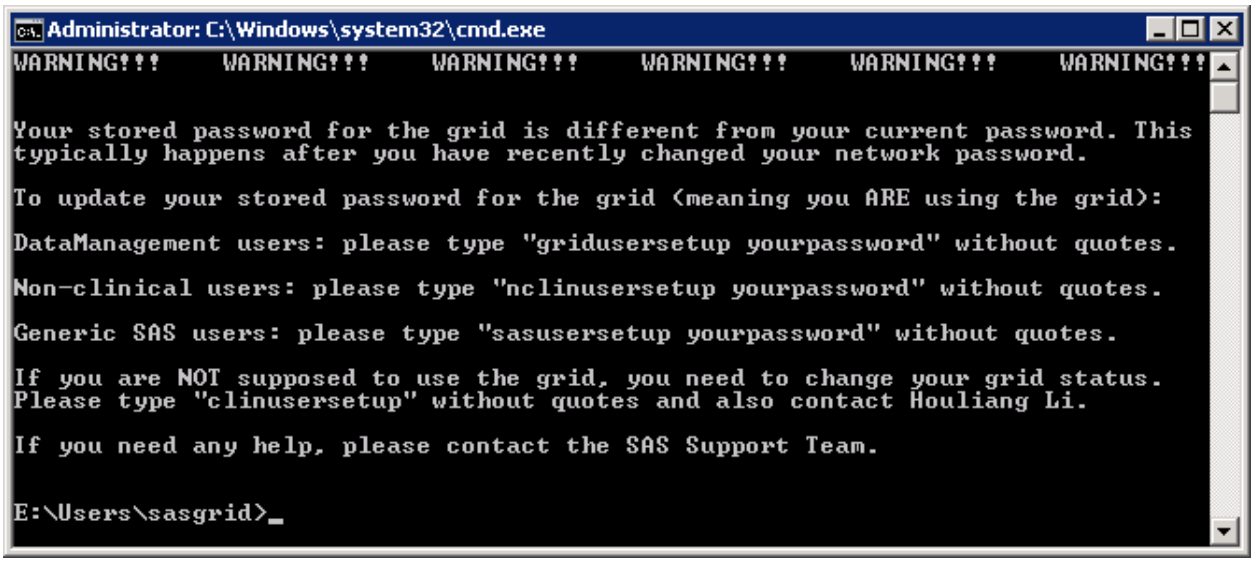

Figure 1: **Automatic Command Prompt for Updating Stored Passwords**

We also made changes to the code sample provided by ***Grid Computing in SAS 9.3***. Instead of either hardcoding or not providing the password in the **`metapass`** option, a **`%sysget`** macro call is used to retrieve the encoded password that has been stored in the user environment variable. Given the unique sequence of launching SAS for development, testing and production at our company, we made this grid access piece part of a macro that is already invoked (via a simple **`%include`** statement) by every study-specific **`autoexec`** file. This means no user is ever prompted for a password before accessing the grid, regardless of launching interactive or batch jobs. For production accounts which have non-expiring passwords, running the batch program once is good forever.

Of course, we have not neglected those casual SAS users who do not need access to the production file system. They can launch SAS as usual by following the Windows Start menu and get to the grid, since the default **`autoexec`** file takes care of the password business.

OPPORTUNITY TWO – MAP ALL THE NETWORK DRIVES

For the grid session to be productive, it needs to be able to access the network storage for the required files and folders. In our company's single-server environment, the local D drive hosted all the production data. We decided to map to the network storage using the same drive letter D on the grid to eliminate the need to change any code, whether SAS programs or batch scripts, that explicitly referenced the drive. When the grid session starts, the very first macro call executed is to map the D drive. Considering that most users also have other network drives mapped in their remote Windows server session, it is only natural to duplicate those drives on the grid node too.

Of course, reality is never this straightforward. When a user connects to the Windows server via Remote Desktop Connection, he or she has the option to make the drives on the user's PC also available to the server session, just like the default printer can be carried over. Since there is no direct connection between any grid node and the user's PC, all such PC-inspired drives need to be filtered out. To further complicate the matter, we have two Windows domains in North America, one current and one legacy. For the drives within the current domain, a short computer name in the UNC path is enough (full name is always good); for the drives within the legacy domain, a fully-qualified computer name is required. This demands that the drive-mapping macro be capable of handling network drives from both domains with variable UNC paths. Actually, the fun does not stop there. We also have two dozen users located on another continent who have their own network domain. When using the grid, some of them need to map to the network drives back in their own country for their programs, which means the drive-mapping macro needs to accommodate their requirements too.

Because mapping network drives can be quite unpredictable due to network setup and traffic, the mapping algorithm has to be robust. It should give enough time for each mapped drive to be connected on the grid node, but it should also back off the attempt if multiple tries have failed. Of course, those drives located in another continent deserve special treatment when being mapped. To optimize end user experience, we eventually created a separate macro just for those users outside of North America. In all cases, if a network drive fails to map within the allotted time window, a NOTE is written to the log to warn the user about it.

OPPORTUNITY THREE – MODIFY DEFAULT BEHAVIORS OF SAS DISPLAY MANAGER

In a regular (i.e., non-grid) SAS Display Manager, a user can either click the **Running-Man** icon on the task bar or use a keyboard shortcut to submit code for execution. In the grid environment, the same user now has two SAS sessions to choose from. At our company, SAS users have always been free to set up their own keyboard shortcuts, so we want to preserve that tradition. In fact, some users find those shortcuts increase their productivity. But we also want to push our users to the grid as much as possible, so we reprogrammed the **Running-Man** icon to make it submit code to the grid session by default. At the same time, we added a new icon, **Running-Man-In-A-Box**, that submits code to the local session. This ensures that users who are not as familiar with keyboard shortcuts also have a way to use the local session if they need to. While temporary data sets created by the local session automatically appear in the WORK library of the SAS Display Manager, it is not easy to see the temporary data sets created in the grid session, i.e., when the **Running-Man** icon is clicked. To facilitate code development and testing, we created a new library, WORKGRID, which points to the WORK library of the grid session. Users can simply click on WORKGRID to access the remote session's temporary data sets.

Because each grid session's host node, picked by the SAS Grid Manager, was only optimal when it was initially selected, the session may not work very well if the grid node later becomes busy. To enable the user to choose another grid node without closing the current SAS Display Manager, we created a **Double-Running-Man** icon that starts a new grid session when clicked. To properly disconnect from a grid session, we also added another icon, the **Stop-Sign**, which terminates the current grid session when clicked. See **Figure 2**. We made sure to add informative tooltips to the new or modified icons as an easy and instant reminder of their proper functions.

Figure 2: Customized Icons

However, our nicely customized icons only applied to the very first SAS session each user started on the server; any subsequent SAS sessions got their profile from the SASHELP library, not the SASUSER library which holds our customizations. You can see this SAS default behavior whenever you launch a second (or third) SAS session and get a few WARNINGs and NOTEs about your user registry and profile. We could not allow the SAS default to defeat our grid intention, so we inserted code to the macro (that is included by the **`autoexec`** file) to restore the customizations to every additional SAS session that is interactive. Batch sessions do not need the customized icons.

OPPORTUNITY FOUR – WORK AROUND / WITH SAS GRID

The **`sasgrid.cmd`** script is the heart and soul of the SAS grid architecture. Because SAS Institute developers have to accommodate a broad spectrum of users and usage scenarios, they have to make certain assumptions and strike a delicate balance when building the grid software. Indeed, if we were a SAS Enterprise Guide shop, everything probably would have worked perfectly out of the box. But we are unique, as many SAS customers are. Through trial and error, we figured out ways to negate certain default behaviors of the grid, such as passing the **`autoexec`** file to the grid session during initial signon, which causes an immediate error when the expected network drive is not available yet, or the ghosting of the SAS initial log messages, which displays the SAS copyright and other startup notes TWICE. We also found ways to work around other problems, such as designating a default printer to enable SAS ODS processing on the grid nodes, as well as preventing the creation of empty "work" folders in our file system.

Throughout the rollout process, we as the SAS administrators continued to be amazed by the variety of ways our experienced and smart SAS users interact with the SAS application. They have so many different methods of launching SAS, some unorthodox but always very effective. In addition to the usual customized SAS icons on the

desktop and targeted Windows batch scripts, some users launch their SAS sessions vie VB scripts tied to an Excel spreadsheet, while others leverage Windows ActiveX and JavaScript to make SAS do impossible things for them. Of course, everyone knows how to use the standard routine based on the Command Prompt. It took us some time and efforts to eventually get all the methods to work with the grid, except the customized SAS icons. It turned out there is some defect in the SAS software itself. Our wonderful SAS Technical Support folks provided us with a fix for this, but we ended up creating a batch alternative that has proven very successful.

DISCUSSION

It is probably true that people don't like changes, especially when the changes do not provide direct benefits to them. In the case of SAS version upgrade or migration to the grid, most users do not really care. As long as the SAS environment performs well and allows them to do their work, they are happy to use the same environment forever. Literally! Learning curves, however small, will be dreaded if the SAS users are already very busy. The inherent uncertainties and inevitable new behaviors ("new features"?) introduced by new versions or architecture only serve to make matters worse, in fact, much worse. On the other hand, the typical corporate IT department is constantly worried about the overstressed and / or deteriorating infrastructure supporting business applications such as SAS. They want to be proactive and to preempt any business-disrupting incidents. They may also want to be prepared for the upcoming surge of business demands as communicated by the business side or upper management.

Our experience of supporting the SAS user community prior to the grid project told us that we should aim to minimize changes imposed on the users. Where changes are absolutely inevitable, we should still strive to mitigate their negative impact on our users or their productivity. In hindsight, we benefited tremendously from the relatively long period of rollout. We were able to put together a basic framework, with password handling and drive mapping functions, etc., before we opened up the test environment to our selected testers. Over three months of focus group testing, we uncovered various problems and hidden issues, and we also received a lot of great suggestions. Fortunately, we had plenty of time to find solutions independently or in consultation with SAS Technical Support.

Before we formally rolled out the production SAS grid 9.3 to our user community, we conducted targeted training sessions specific to each user group's needs. Training materials, detailed instructions, and useful tips and tricks were communicated to all users in a timely manner. When the formal rollout began on a Monday morning, we were delighted to see that almost all our SAS users went right in to the grid and continued their work from the previous week, as if nothing had happened. The grid deployment has not caused any business disruptions, and our company is now ready for the future as far as the SAS environment is concerned.

CONCLUSION

Deploying SAS grid is a big challenge in any user community. It is even more difficult if your users primarily rely on SAS Display Manager and cannot get out of it. With careful planning, thorough testing, an adequate rollout window, and above all, innovative solutions to all the problems presented by both the grid architecture and your unique circumstances, you can make the experience an almost transparent one. Your users should feel very little change, if at all, in how they interact with SAS.

As I often like to say to our users, if they already know SAS/Connect, great; they can continue to utilize their knowledge on the grid if they like. However, if they have never learned to use SAS/Connect before, they are not missing much either. In our grid environment, no SAS user ever needs to type **`rsubmit`** or **`endrsubmit`** to access the grid. They don't have to learn about SAS/Connect; they have customized icons, with ready tooltips, to guide them to the grid and back. When all is said and done, your users may not call you a hero (some do, actually!), but you know in your heart that you have saved their world. What is in a name, right?

ACKNOWLEDGMENTS

The author would like to express his sincere gratitude to colleagues Pete Lineman and Dennis Snyder for their steadfast support throughout the grid deployment. It has been such a great pleasure to work with the best project manager (Pete) and the most effective systems administrator (Dennis). What a tremendous team!

CONTACT INFORMATION

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

Houliang Li

Frederick, MD

(301) 693-3722

Houliang_Li@yahoo.com